

Integrity

Utilities Guide

Version 5.0 Rev. 0



Integrity Utilities Guide, Version 5.0 Rev. 0

July 2014

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) 1985-2014 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: DOC108-500

Contents Summary

Preface xxi

Part I OPAC Blocks Reference 1

Chapter 1 Summary of OPAC Blocks 3

Chapter 2 General Action Palette 19

Chapter 3 Decisions Palette 57

Chapter 4 Operating System (OS) Palette 85

Chapter 5 Stack Operations Palette 103

Chapter 6 Local Parameters Palette 115

Chapter 7 Subtask Arguments Palette 131

Chapter 8 Debugging Palette 137

Chapter 9 State Transition Palette 141

Chapter 10 External Interfaces Palette 159

Chapter 11 Generic Blocks Palette 175

Chapter 12 Message Palette 185

Part II Utilities 205

Chapter 13 OpEx Dispatch Engine Reference (ODIE) 207

Chapter 14 Message Parsing Engine (MPE) 275

Part III Autodiscovery 333

Chapter 15 IP Reachability Analyzer (IPRA) 335

Chapter 16 Object Reachability Analysis (ORA-TWO) 345

Chapter 17 Domain Export/Import (DXI3) 359

Chapter 18 Open View Map Importer (OVMAP) 377

Chapter 19 Ping Manager 389

Part IV G2-SNMP Bridges 397

Chapter 20 Overview of the G2-SNMP Bridges 399

Chapter 21 Installation and Startup 407

Chapter 22 G2-SNMP Bridge Setup 423

Chapter 23 G2-SNMP Bridges API 455

Chapter 24 Reporting Errors 507

Part V APIs and Initializations 509

Chapter 25 Core Services APIs 511

Chapter 26 OPAC APIs 533

Chapter 27 Startup Parameters 541

Glossary 551

Index 555

Contents

Preface xxi

About this Manual xxi

Version Information xxii

Audience xxii

Note to Integrity Users xxii

A Note About the API xxiii

Conventions xxiii

Related Documentation xxv

Customer Support Services xxvii

Part I **OPAC Blocks Reference** 1

Chapter 1 **Summary of OPAC Blocks** 3

Introduction 3

General Actions Palette 4

Decisions Palette 7

OS Actions Palette 9

Stack Operations Palette 10

Local Parameters Palette 11

Subtask Arguments Palette 13

Debugging Palette 13

State Transition Palette 14

External Interfaces Palette 15

Generic Blocks Palette 16

Message Palette 17

Chapter 2 General Action Palette 19

Introduction	20
New Procedure	21
General Procedure	22
Send SMH Message	24
Historical Message Query	27
Hide Workspace	30
Show Workspace Not Stacked	32
Show Workspace	34
Block Pause Capability	36
Control Delay	38
Task Kill	39
Task Spawn	40
Subtask	42
Macro	44
Subtask Completion	46
Subtask Start	47
Procedure Statement	49
Procedure Template	53
Connection Post	56

Chapter 3 Decisions Palette 57

Introduction	58
Comparison Decision	59
2-Way Decision	62
2-Way Manual Decision	66
3-Way Manual Decision	69
4-Way Manual Decision	71
2-Way Pattern Decision	73
3-Way Pattern Decision	76
4-Way Pattern Decision	78

	2-Way Pattern Decision By Symbol	81
Chapter 4	Operating System (OS) Palette	85
	Introduction	85
	Set Local Integer From Source	87
	File Exists Test	88
	Delete File	90
	Kill Process	92
	Spawn Return Output	94
	Spawn Return PID	96
	Spawn No Return	98
	Write File	99
	Read File	101
Chapter 5	Stack Operations Palette	103
	Introduction	103
	Generic Put Something On Stack	105
	Pop General Stack	107
	Put Connected Objects On Stack	108
	Pop General Stack And Delete	110
	Put Item On Stack	111
	Put Float On Stack	112
	Put Integer On Stack	113
	Put Text On Stack	114
Chapter 6	Local Parameters Palette	115
	Introduction	116
	Set Local Float From Source	117
	Local Float Parameter	119
	Set Local Item From Source	121
	Local Item	122
	Set Local Integer From Source	124

	Local Integer Parameter	125
	Set Local Text From Source	127
	Local Text Parameter	128
Chapter 7	Subtask Arguments Palette	131
	Introduction	131
	Value Argument	132
	Item Argument	134
Chapter 8	Debugging Palette	137
	Introduction	137
	Show Stack Top	138
	Show Token Info	139
Chapter 9	State Transition Palette	141
	Introduction	141
	State Transition Diagrams	143
	Delete State Token	148
	Get State	149
	Accept New Event	151
	Accept New State	153
	State Diagram Completion	154
	Transition Event	155
	Wait State	156
	State Diagram Start	157
Chapter 10	External Interfaces Palette	159
	Introduction	160
	Read Domain Map	161
	Write Domain Map	164
	SNMP Get Table Column	166
	SNMP Set	168
	SNMP Get	170

Send CDG Event 172

Chapter 11 Generic Blocks Palette 175

Introduction 175

Get Related Items 176

Historical Numerical Query 178

Iteration 180

Run Domain Object Method 182

Set Local Parameter From Source 184

Chapter 12 Message Palette 185

Introduction 185

Send Message 187

Current Message Query 190

Clear Message History 193

Message Exists 195

Set Message Attribute 197

Delete Message 200

Acknowledge Message 202

Part II Utilities 205

Chapter 13 OpEx Dispatch Engine Reference (ODIE) 207

Introduction 208

Events 209

Publish Subscribe Mechanism 209

Managers 210

Subscribers 210

Old Event Processing 210

Filters 211

Target Class Filter 211

Target Attribute Filter 211

Delay Filter 211

Time Filter 211

- Query Filter **211**
- Attribute Filter **212**
- Hour of the Day Filter **212**
- Day of the Week Filter **212**
- Message Historical Query Filter **212**
- Message Query Filter **213**
- Event Count by Start Time **213**
- Passport Filter **213**
- Event Class Filter **214**
- Making Your Own Filter Block **214**

Responses **214**

- Delete Event **214**
- Delete Events by Start Time **215**
- G2 Procedure Response **215**
- Create Message **215**
- Clears for or Delete Messages **216**
- Delete Message **216**
- Acknowledge Message **216**
- Beep **216**
- Log_Event **216**
- Starting an OPAC Procedure **217**
- Using Indirect References **218**

OPAC Blocks for ODIE Events **218**

- Publish New Event **218**
- Publish Event **218**
- Delete Event **219**
- Delete Events **219**
- Get Event Attribute **219**
- Set Event Attribute **219**
- Add Passport to Event **220**
- Count Events **220**
- Gather Evidence **220**
- Using Indirect References **221**

Subscriber Toolbox **221**

Classes **223**

- odie-event **224**
- odie-event-proxy **226**
- odie-g2-manager **228**

Application Programmer's Interface **230**

- odie-g2-manager::odie-datastore-add-event-passport-stamp **232**
- odie-g2-manager::odie-datastore-create-event **233**
- odie-g2-manager::odie-datastore-delete-event **235**
- odie-g2-manager::odie-datastore-delete-events **236**
- odie-g2-manager::odie-datastore-duration-count-query **238**

Synopsis 238

- odieg2manager::odie-datastore-duration-proxy-query 240
- odieg2manager::odie-datastore-get-event-attribute-value 242
- odieg2manager::odie-datastore-get-passport-stamps 243
- odieg2manager::odie-datastore-set-event-attribute-value 244
- odieg2manager::odie-datastore-start-time-count-query 245
- odieg2manager::odie-datastore-start-time-proxy-query 247
- odie-manager::odie-manager-add-event-passport-stamp 249
- odie-manager::odie-manager-create-event-class 250
- odie-manager::odie-manager-delete-event 251
- odie-manager::odie-manager-delete-events 252
- odie-manager::odie-manager-duration-count-query 254
- odie-manager::odie-manager-duration-proxy-query 256
- odie-manager::odie-manager-get-event-attribute-value 258
- odie-manager::odie-manager-get-passport-stamps 259
- odie-manager::odie-manager-passport-meets-include-exclude-criteria 260
- odie-manager::odie-manager-post-inform-statement 261
- odie-manager::odie-manager-publish-existing-event 262
- odie-manager::odie-manager-publish-new-event 263
- odie-manager::odie-manager-publish-new-event 264
- odie-manager::odie-manager-set-event-attribute 266
- odie-manager::odie-manager-start-time-count-query 267
- odie-manager::odie-manager-start-time-proxy-query 269
- odie-manager::odie-manager-subscribe-event-class 271
- odie-manager::odie-manager-substitute-attribute-values 272
- odie-manager::odie-manager-unsubscribe 273
- odie-manager::odie-manager-unsubscribe-event-class 274

Chapter 14 Message Parsing Engine (MPE) 275

Introduction 276

General Information 276

The OMPE String Receiver 276

Message Filter 278

Message Parsing Engine Palette Blocks 279

Conclude Blocks 279

Procedure Conclude 279

Single Regex Conclude 280

Start End Of Text Conclude 280

Start End Regex Conclude 280

Start Of Text To End Of Regex Conclude 281

Static Conclude 281

String Position 281

String Receiver 281

Word Line 283

Debug Blocks	284
Pause	284
Decision Blocks	284
Single Match Decision	284
Start End Of Match Decision	284
Integrity Subsystem Blocks	285
Create Message	285
Delete Message	285
Opac Subtask Start	286
Message Handling	287
Message Filter	287
Text Buffer	287
Terminal Blocks	287
Terminal	287
Classes	288
mpe-message-filter	289
mpe-pause-block	292
mpe-procedure-conclude-block	294
mpe-single-match-decision-block	296
mpe-single-regex-conclude-block	298
mpe-start-end-match-decision-block	300
mpe-start-end-of-text-conclude-block	302
mpe-start-end-regex-conclude-block	304
mpe-start-of-text-to-end-regex-conclude-block	306
mpe-static-conclude-block	308
mpe-string-position-block	310
mpe-string-receiver	312
mpe-terminal-block	313
mpe-text-buffer	314
mpe-word-line-conclude-block	316
create-message-block	318
ompe-delete-message-block	320
ompe-opac-subtask-start-block	322
ompe-string-receiver	324
Application Programmer's Interface	326
mpe-current-real-time-as-time-stamp	326
mpe-text-buffer::mpe-add-text-to-buffer	327
mpe-text-buffer::mpe-clear-buffer	327
User Menu Choices	328
mpe-clear-buffer	328
mpe-show-buffer	328
mpe-turn-debugging-off	328
mpe-turn-debugging-on	329
ompe-go-to-procedure	329
Relations	330

`_mpe-from-message-filter` 330
`_mpe-from-text-buffer` 330

Part III Autodiscovery 333

Chapter 15 IP Reachability Analyzer (IPRA) 335

Introduction 335
Setting up G2/IPRA 336
Setting Up the Ping Manager 338
 Troubleshooting an IPRA Ping Manager GSI-Interface 339
Summary of IPRA Default Behavior 340
Procedures 341

Chapter 16 Object Reachability Analysis (ORA-TWO) 345

Introduction 346
Concepts 346
 Node Types 346
 Polling 347
Setup 347
Manager Object 348
Event Methods 350
Domain Methods 351
Support Procedures 354
Additional Procedures 356
Report Procedures 357

Chapter 17 Domain Export/Import (DXI3) 359

Introduction 360
Integrity Export Import Toolbox 360
The DXI3-file Format 360
 Remarks on the Syntax 361
A “Bad” Import File and Data Corruption 363
 Types and Handling of 'Bad' Data and DXI3 363
 Errors Particular to the dxi3-import File 363
 Errors Common to the dxi3 API and File Use 363

Effects of 'Bad' Data on the Domain Map	364
Type to Class Mapping	364
Containment and Other Types of Hierarchies	365
Exporting Domain Maps	365
Importing a Domain Map	368
Example	369
The Example Network	370
The Data Structure	370
Notes/Assumptions	371
DXI3 APIs	372
dxi3-register-domain-item	373
dxi3-register-domain-relation	374
dxi3-register-domain-attribute-value	375
Format	375
Chapter 18 Open View Map Importer (OVMAP)	377
Introduction	377
System Requirements	378
Installation	378
Network Account Setup	378
Ovobjprint Command	379
Testing	379
Installation of Modules	380
Setup of Incremental Addition of Domain Objects	380
Detailed Descriptions	381
Class Definitions	381
dxi3-import-object	381
dxi3-type-to-class-object	383
Translations	384
Initializations	384
File Transfer Routines	385
Translation Objects	385
New Class Creation	386
OV Map Importer Operation	387
Building the Domain	387
Incremental Build	387
Notes on GDXI	388

Chapter 19 Ping Manager 389

Introduction 389

Components 390

Running the Ping Manager 390

The Remote Procedure Calls 391

 Setting the Device Configuration for a Ping Manager 391

 Changing a Device's Configuration for the Ping Manager 392

Application Development 394

 Demand Polling 394

 Periodic Polling 394

Sample Procedures and Actions for pingmgr.kb 394

 A Sample Configuration File 394

 Example 395

 Example of a Procedure to a get configuration status 395

 Example of an Action-Button to Invoke get-device-status 396

Part IV G2-SNMP Bridges 397

Chapter 20 Overview of the G2-SNMP Bridges 399

Introduction 399

Applications 401

Features and Benefits 401

Acquiring Data 401

Building a G2-SNMP Bridge Application 402

G2-SNMP Bridges and the Integrity Product Family 403

Enhancements 404

Chapter 21 Installation and Startup 407

Introduction 407

UNIX Platform Installation 407

 Installing from Tape 407

 Determining the Device Name 408

 G2-SNMP "Generic" Bridge Additional Installation Steps 409

 Installing from CD-ROM 411

 G2-SNMP "Generic" Bridge Additional Installation Steps 414

Authorizing the G2-SNMP Bridges 415

Authorizing the SNMP Gateway Bridge	415
Executing the G2-SNMP Bridge	416
Executing the SNMP Gateway Bridge	416
Finding an Available Port	417
Running SNMP Gateway Bridges as Background Processes	418
Executing the Integrity Application	418
Connecting G2 to the GSI Bridge Process	419
Creating a GSI Interface Object	419
Configuring the GSI Interface Object	420
Chapter 22 G2-SNMP Bridge Setup	423
Introduction	424
Configuring the G2-SNMP Bridge	424
SNMP Gateway Bridge Configuration	425
Communication Parameters	425
Filtering Traps	425
Telling the SNMP Gateway Bridge Which Traps to Filter	426
Error Handling	426
Creating a New Error Handling Procedure	427
Trap Handling Overview	428
Trap Class Creation and Processing	430
Handling Unrecognized SNMP Traps	431
SNMP Traps	432
Trap Manager	432
Trap Properties	432
Defined Trap Properties	433
Trap Processing	433
MIB Processing	434
Setting Up and Running the MIB Parser	434
MIB Parser Setup	434
Processing MIB Files	434
Viewing a Parsed MIB	435
Installed MIBs	435
Vendor MIBS	436
trapd.conf.ppd Parser	437
Clears-For Attribute	439
Completion Procedure Determination	441
SNMP Transactions	442
Blocking and Non-Blocking Transactions	443
Overloading Remote Procedures	443
Sending Traps to External Systems	443
HP OpenView Interface	443

	Sending an HP OpenView status trap	443
	NetView 6000 Interface	444
	Simulation Facilities	445
	SNMP Trap Simulation	445
	SNMP Agent MIB Simulation	450
Chapter 23	G2-SNMP Bridges API	455
	Introduction	456
	Update for GSI-Based Bridge Process	456
	Support for Filtering of Traps from Specified Hosts	456
	Passing Variable Values for Variable Bindings in Which the Variable Type Is 'Object Identifier'	457
	Remote Procedure Calls	458
	Base RPCs	458
	g2snmp_add_filtered-trap	458
	g2snmp_delete_filtered_trap	461
	g2snmp_modify_comm_params	463
	g2snmp_set_agent_filter_mode	463
	g2snmp_use_snmp_comm_params	464
	g2snmp_use_snmp_defaults	464
	g2snmp_blocking_transaction	465
	g2snmp_nonblocking_transaction	467
	Overloaded RPCs	468
	get_nonblocking_single	468
	get_blocking_single	469
	get_2_blocking	470
	set_blocking	472
	set_nonblocking_integer	473
	set_nonblocking_text	474
	send_novar_trap_nonblocking	475
	send_trap_nonblocking	477
	send_trap_status_nonblocking	479
	Receiver Procedures	481
	g2snmp_receive_eot	481
	g2snmp_receive_float	482
	g2snmp_receive_integer	483
	g2snmp_receive_message	483
	g2snmp_receive_string	484
	g2snmp_receive_trap_packet	485
	Procedures Listed by Module	488
	GNDO Module	488
	GMIB Module	491
	GSNMP Module	495

	Functions	503
Chapter 24	Reporting Errors	507
Part V	APIs and Initializations	509
Chapter 25	Core Services APIs	511
	Introduction	511
	Procedures Listed by Module	512
	GNDO Module	513
	GLF Module	529
	Functions	530
	Methods Listed by Class	532
Chapter 26	OPAC APIs	533
	Introduction	533
	External APIs Calling OPAC from G2	534
	Internal APIs for User-Written Blocks	534
	Other Utility API's for User-Written Blocks	537
	OPAC Error Handling	538
	State Transition Diagram APIs	538
	Debugging OPAC Procedures	540
Chapter 27	Startup Parameters	541
	Introduction	541
	GNDO Module	542
	Error Handling:	542
	Object Retrieval	543
	Colors and Priority	544
	GMIB Module	546
	GSNMP Module	546
	Global Parameters	547
	Performance Parameters	548
	MIB Module	548

GSNMP Module 549

Glossary 551

Index 555

Preface

Describes this document and the conventions that it uses.

About this Manual	xxi
Version Information	xxii
Audience	xxii
Note to Integrity Users	xxii
A Note About the API	xxiii
Conventions	xxiii
Related Documentation	xxv
Customer Support Services	xxvii



About this Manual

This manual contains reference information about the graphical components (palette blocks) APIs, and initializations of the Core Services and the OPERator ACtions (OPAC) components of the Integrity product family.

This manual is designed to help users quickly develop and deploy applications that monitor and control networks and the systems, services, and applications that run on them. This manual:

- Provides a detailed reference of [OPAC blocks](#), which are the core objects used to create an Integrity application.
- Describes the following modules:
 - OpEx Dispatch Engine Reference (ODIE) – A tool for handling events and responses to events.

- Message Parsing Engine (MPE) – A graphical language specifically for parsing text messages.
- Data Point Integration – Tools for importing and exporting domain maps from and to an SQL-compliant database.
- IP Reachability Analyzer (IPRA) – A starting point for providing reachability analysis, using other products that are part of the Integrity family of products.
- Object Reachability Analysis (ORA-TWO) – Provides root cause reachability analysis for any ‘threaded’ network.
- Domain Export/Import (DXI3) –
- Open View Map Importer (OVMAP) – Provides tools for importing and exporting domain objects.
- Ping Manager – Provides tools for testing the communication path from a sender to receiver via the Internet.
- Describes the [SNMP Bridges](#), which enable a user application to communicate with devices that support the Simple Network Management Protocol (SNMP).
- Describes the Integrity [application programmers’ interface and initializations](#).

Version Information

Integrity requires Gensym’s G2 to run. Integrity OPAC also requires the use of Gensym’s Integrity Core Services product.

Audience

The assumed audience for this guide is applications developers, who have basic experience in programming computerized applications. This document also assumes that developers are familiar with G2 terminology and operations, but it does not require a thorough understanding of G2. If you encounter G2 terms or concepts that you do not understand, refer to the *G2 Reference Manual*.

Note to Integrity Users

Integrity demonstrations include object classes for use in network management applications. However, Integrity applications in general are in no way restricted to use in network management.

Good candidates for Integrity applications include any system containing alarms and events in general that must be analyzed, managed, and possibly correlated by

using information contained in schematics, where procedures must be automated and tests run.

A Note About the API

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

The techniques by which Integrity implements its capabilities, however, are subject to change at any time without notice or explanation, and are expected to change as Integrity evolves. These techniques will not be described in any Integrity documentation.

Therefore, it is essential that you use Integrity exclusively through its API, as described in this Integrity manual. If you bypass the API, you cannot rely on your code to work in the future, since Integrity may change, or in the present, because the code may not correctly manage the internal operations of Integrity.

Conversely, if you use the Integrity API exclusively, you can rely on Gensym to notify you of any Integrity changes that might affect your code, and you can rely on Integrity to manage all internal operations correctly.

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

OPAC Blocks Reference

Chapter 1: Summary of OPAC Blocks

Provides a summary of the blocks in the OPAC toolbox.

Chapter 2: General Action Palette

Provides descriptions, configuration dialog box and possible attribute values for each item on the OPAC General Actions palette.

Chapter 3: Decisions Palette

Provides descriptions, configuration dialog box and possible attribute values for each block on the Decisions palette.

Chapter 4: Operating System (OS) Palette

Provides descriptions, configuration dialog box and possible attribute values for each item on the OPAC OS Actions palette.

Chapter 5: Stack Operations Palette

Provides a description, the configuration dialog box, and possible attribute values for each block on the OPAC Stack Operations palette.

Chapter 6: Local Parameters Palette

Provides a description, the configuration dialog box, and possible attribute values for each item on the OPAC Stack Local Parameters palette.

Chapter 7: Subtask Arguments Palette

Provides a description, the configuration dialog box, and possible attribute values for each item on the OPAC Subtask Arguments palette.

Chapter 8: Debugging Palette

Provides a description, the configuration dialog box, and possible attribute values for each item on the OPAC Debugging palette.

Chapter 9: State Transition Palette

Provides descriptions, configuration dialog box and possible attribute values for each item on the State Transition palette.

Chapter 10: External Interfaces Palette

Provides a description, the configuration dialog boxes, and possible attribute values for each item on the External Interface palette.

Chapter 11: Generic Blocks Palette

Provides descriptions, configuration dialog box and possible attribute values for each item on the Generic palette.

Chapter 12: Message Palette

Provides descriptions, configuration dialog box and possible attribute values for each item on the Message palette.

Summary of OPAC Blocks

Provides a summary of the blocks in the OPAC toolbox.

Introduction	3
General Actions Palette	4
Decisions Palette	7
OS Actions Palette	9
Stack Operations Palette	10
Local Parameters Palette	11
Subtask Arguments Palette	13
Debugging Palette	13
State Transition Palette	14
External Interfaces Palette	15
Generic Blocks Palette	16
Message Palette	17







Introduction








The Integrity development environment provides toolboxes to allow a developer to quickly and easily build or modify an application. Blocks are grouped into palettes and palettes are grouped in toolboxes. This chapter provides a guide to the toolboxes that are included in Integrity.






Blocks are arranged in one of these palettes:

- [General Actions Palette](#)
- [Decisions Palette](#)
- [OS Actions Palette](#)
- [Stack Operations Palette](#)
- [Local Parameters Palette](#)
- [Subtask Arguments Palette](#)
- [Debugging Palette](#)
- [State Transition Palette](#)
- [External Interfaces Palette](#)
- [Generic Blocks Palette](#)
- [Message Palette](#)

General Actions Palette







Palette Block	Description
OPAC-New-Procedure 	Assists you in creating a user-defined OPAC procedure by providing a container workspace preconfigured with an OPAC-Subtask-Start block and an OPAC-Subtask Completion block.
OPAC-General-Procedure 	Use to embed your own actions within an OPAC procedure.
Hide Workspace 	Hides the specified workspace.
OPAC-Show-Workspace- NOT-Stacked 	Show the specified workspace and do not place it on the general stack of the token.







Palette Block	Description
OPAC-Show-Workspace 	Show the specified workspace and place it on the general stack of the token.
OPAC-Block-Pause-Capability 	Place on a workspace to turn token movement display on. Connect to blocks on a workspace to configure procedure execution pause conditions and generate dialog for operator action to continue or abort procedure.
OPAC-Control-Delay 	Specify timed delay in a procedure.
OPAC-Task-Kill 	Abort processing and delete token.
OPAC-Task-Spawn 	Start a new procedure with a new token.
OPAC-Subtask-Block 	Call a subtask, transferring control to the subtask until the subtask procedure is complete. Arguments can be passed to the subtask.
OPAC-Macro 	Call a macro, transferring control to the subtask until the macro procedure is complete. Arguments cannot be passed to the macro. The macro can use local parameters created within the macro. Local parameters in the calling procedure are also available to the macro.

Palette Block	Description
OPAC-Subtask-Completion 	Indicate the end of an OPAC procedure.
OPAC-Subtask-Start 	Indicate the beginning of an OPAC procedure. The name attribute of this block identifies the subtask.
OPAC-Procedure-Statement 	Allows you to embed any G2 action into an OPAC procedure.
OPAC-Procedure 	Create a new OPAC procedure template.
Connection-Post 	Allows you to connect OPAC procedures across workspaces.







Decisions Palette



Note All decision blocks delete the item at the top of the stack.

Palette Block	Description
OPAC-2-Way-Decision 	Provides two-way branching capability. Branching is based on a user-defined G2 procedure.
OPAC-2-Way-Manual-Decision 	Prompts an operator to decide to continue processing between two branches, allowing specification of a timeout default decision branch.
OPAC-2-Way-Pattern-Decision 	Provides two-way branching capability. Branching is based on the comparison of the items at the top (or beginning) of the token stack against the patterns specified in the choice 1 and choice 2 attributes, respectively.
OPAC-2-Way-Decision-By-Symbol 	Provides two-way branching capability. Branching is based on specified parsing of an input string into one or more symbols.
OPAC-3-Way-Decision 	Provides three-way branching capability. Branching is based on a user-defined G2 procedure.
OPAC-3-Way-Manual-Decision 	Prompts an operator to decide to continue processing among three branches, allowing specification of a timeout default decision branch.





Palette Block	Description
OPAC-3-Way-Pattern-Decision 	Provides three-way branching capability. Branching is based on the comparison of the token against the patterns specified in the choice 1, choice 2, and choice 3 attributes, respectively.
OPAC-4-Way-Decision 	Provides four-way branching capability. Branching is based on a user-defined G2 procedure.
OPAC-4-Way-Manual-Decision 	Prompts an operator to decide to continue processing among four branches, allowing specification of a timeout default decision branch.
OPAC-4-Way-Pattern-Decision 	Provides four-way branching capability. Branching is based on a comparison of the token against the patterns specified in the choice 1, choice 2, choice 3, and choice 4 attributes, respectively.
OPAC-Comparison-Decision 	Provides two-way branching capability. Branching is based on evaluation of a relational expression that evaluates to True or False.
OPAC-Decision-Procedure 	Provides a procedure for 2-Way, 3-Way, and 4-Way Decision blocks.





OS Actions Palette

Palette Block	Description
OPAC-File-Exists-Test 	Determines if the specified file exists.
OPAC-Delete-File 	Deletes the specified file.
OPAC-Kill-Process 	Kills an operating system process identified by the Process ID (PID) that is specified by the block's Process ID Source attribute. Valid sources are the token stack or a local parameter.
OPAC-Spawn-Return-Output 	Spawns an operating system process and returns the output of the process to the stack.
OPAC-Spawn-Return-PID 	On a Unix system, spawns an operating system process and returns the process ID (PID) assigned to the process by the operating system to the specified source. (In Windows, only the command shell PID of the process is returned.)
OPAC-Spawn-No-Return 	Spawns a process and returns no value. Equivalent to a command line command.




Palette Block	Description
OPAC-Write-File 	Writes the text at the top of the token stack to the specified file.
OPAC-Read-File 	Reads a file and places the result on the token stack.







Stack Operations Palette

Palette Block	Description
OPAC-Generic-Put-Something-On-Stack 	Puts something on the stack that is user-defined in the G2 User Procedure named in the block's User Procedure attribute.
OPAC-POP-General-Stack 	Removes the top item from the stack.
OPAC-Put-Connected-ObjectS-On-Stack 	Locates all objects connected to the object specified, lists them on the stack.
OPAC-POP-General-Stack-And-Delete 	Removes the top item from the stack, and deletes it.



Palette Block	Description
OPAC-Put-Item-On-Stack 	Places the specified item on the top of the token stack. Substitution variables are allowed in the specification.
OPAC-Put-Float-On-Stack 	Places the specified floating decimal value on the top of the token stack. Substitution variables are allowed in the specification.
OPAC-Put-Integer-On-Stack 	Places the specified integer on the top of the token stack. Substitution variables are allowed in the specification.
OPAC-Put-Text-On-Stack 	Places the specified text value and places on the top of the token stack. Substitution variables are allowed in the specification.

Local Parameters Palette



Palette Block	Description
OPAC-Local-Float-Parameter 	Defines a local float parameter for the attached OPAC-Subtask-Start block.
OPAC-Local-Integer-Parameter 	Define a local integer parameter for the attached Subtask Start block.
OPAC-Local-Item 	Defines a local item parameter for the attached OPAC-Subtask-Start block.

Palette Block	Description
OPAC-Local-Text-Parameter 	Defines a local text parameter for the attached Subtask Start block.
OPAC-Set-Local-Float-From-Source 	Sets the specified local float parameter to the value specified by the source.
OPAC-Set-Local-Integer-From-Source 	Sets the specified local parameter to the integer specified by the source.
OPAC-Set-Local-Item-From-Source 	Sets the specified local parameter to the value specified by the source. If the stack is empty, the Local Item is set to the token.
OPAC-Set-Local-Parameter-From-Source 	Assigns a value to a local parameter from a user-defined G2 procedure.
OPAC-Set-Local-Text-From-Source 	Sets the specified local text parameter to the text specified by the source.








Subtask Arguments Palette



Palette Block	Description
OPAC-Value-Arg 	Passes the specified value as an argument to an OPAC subtask .
OPAC-Item-Arg 	Passes a value referencec by the specified item name as an argument to an OPAC Subtask.

Debugging Palette





Palette Block	Description
OPAC-Show-Stack-Top 	Displays the top of the token stack.
OPAC-Show-Token-Info 	Displays information about the token, including \$block, \$caller, \$target, \$window, \$notify, and the current values of any local parameters attached to the Subtask-Start block.



State Transition Palette

Palette Block	Description
OPAC-Delete-State-Token-Block 	Deletes the token associated with the specified target within the specified state diagram.
OPAC-Get-State-Block 	Retrieves the current state for the specified target within the specified state transition diagram and places the result at a specified destination.
OPAC-Accept-New-Event-Block 	Moves the token and specified target to a new state, based on the specified event.
OPAC-Accept-New-State-Block 	Moves the token to a new state, bypassing any transition event blocks.
OPAC-State-Diagram-Completion 	Marks the completion of the processing designated by a State Transition diagram.
OPAC-Timeout-Transition-Event 	Similar to the Transition-Event block, but provides timeout capability. After the timeout, the token transitions to the next connected Wait State.
OPAC-Transition-Event 	Waits for the specified event and then executes the specified Event Action Procedure.





Palette Block	Description
OPAC-Wait-State 	Defines a system state and its associated procedure.
OPAC-State-Diagram-Start 	Begins a state diagram. The state diagram is referenced by the name of this block.



External Interfaces Palette

Palette Block	Description
OPAC-Write-Domain-Map 	Creates an ASCII text file from the top-level object supplied by the user.
OPAC-Read-Domain-Map 	Reads a formatted ASCII text file and creates domain objects and connections based on information in the file and on any specified translation workspace.
OPAC-SNMP-Get-Table-Column 	Gets a table of values for a specified object id from a specified agent hostname.
OPAC-SNMP-Set 	Performs an SNMP Set request to set a value of a single variable.





Palette Block	Description
OPAC-SNMP-Get 	Performs an SNMP Get request to obtain the value of a single variable.
OPAC-Send-CDG-Event 	Sends an event to SymCure for processing.






Generic Blocks Palette

Palette Block	Description
OPAC-Get-Related-Items-Block 	Collects items that are related by a G2 or user-defined relation. Valid relations are connected-to, superior-to, connected-upstream-to, connected-downstream-to, or any user-defined relation. Collected items are placed on the token stack.
OPAC-Historical-Numerical-Query 	Allows access to and calculation of statistics, based on historical numerical data stored in a specified target domain object over a specified time. The specified statistical function performed on the data can be any defined method.
OPAC-Iteration 	Applies a "For Loop," using the specified method to iterate over one of three iteration classes: members of a class, a list of objects, or a range of numbers.
OPAC-Iteration-Procedure-Template 	Used with the Iteration block to allow you to provide your own code for iterating over objects.

Palette Block	Description
OPAC-Domain-Object-Method 	Allows you to create your own general method.
OPAC-Run-Domain-Object-Method 	Run the specified method against the specified object.

Message Palette

Palette Block	Description
OPAC-Acknowledge-Message 	Acknowledges messages based either on specified target, sender, and category; or on a local item variable.
OPAC-Clear-Message-History 	Clears message history specified by indicated target, sender, or categories.
OPAC-Current-Message-Query 	Returns either a list of messages or a count of current messages that match the specified criteria.
OPAC-Delete-Message 	Deletes from the message server the messages identified either on specified target, sender, and category; or on a local item variable. Includes deletion of unacknowledged messages that meet the criteria.

Palette Block	Description
OPAC-Historical-Message-Query	Use to specify a query against the message history.
	
OPAC-Message-Attribute-Procedure-Template	Used in conjunction with the Set Message Attribute block to set attribute values.
	
OPAC-Message-Exists	Determines whether a message exists (on a server) identified by the specified target, sender, and category, and returns a truth value.
	
OPAC-Send-SMH-Message	Create and send a message to a message server.
	
OPAC-Set-Message-Attribute	Sets a specified message attribute to a value.
	

General Action Palette

Describes the blocks in the General Actions palette.

Introduction	20
New Procedure	21
General Procedure	22
Send SMH Message	24
Historical Message Query	27
Hide Workspace	30
Show Workspace Not Stacked	32
Show Workspace	34
Block Pause Capability	36
Control Delay	38
Task Kill	39
Task Spawn	40
Subtask	42
Macro	44
Subtask Completion	46
Subtask Start	47
Procedure Statement	49
Procedure Template	53
Connection Post	56

Introduction

The General Actions palette blocks are described in this chapter. These palette blocks represent the general actions required to create and modify an Integrity OPAC application.

Here is the General Actions palette:



To use the OPAC General Actions Palette blocks, drag and drop blocks to a workspace, then configure the blocks.

- Use General Actions blocks in conjunction with other procedure blocks.
- Use the Subtask Start and Subtask Completion to begin and end all OPAC procedures.

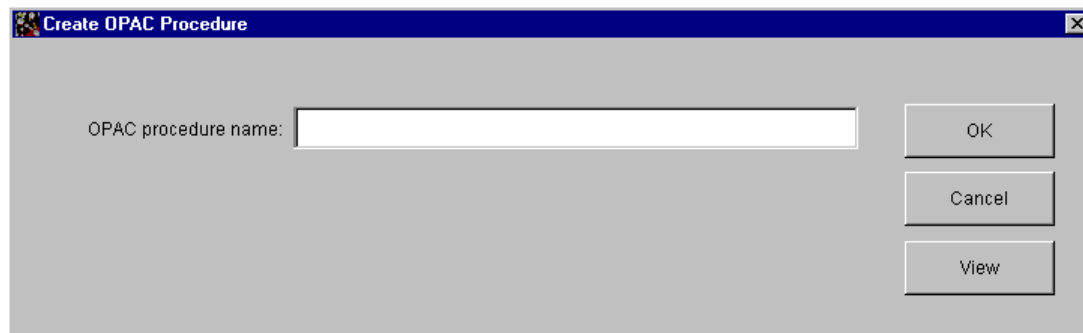
New Procedure



The New Procedure block assists you in creating a user-defined OPAC procedure by providing a container workspace preconfigured with an Subtask Start block and an Subtask Completion block.

To configure the new procedure, right-click on the block and select Properties.

Enter the new OPAC procedure name in the text box, enter an optional description and click on the OK button.

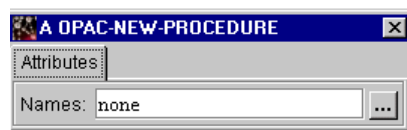


To layout the blocks in the procedure, right-click on the block and select Show Details. The subworkspace of the new procedure appears. The Subtask Start block and the Subtask Completion block are already in place on the subworkspace.

To change the name of the procedure, right-click on the block and select Properties, then enter the new name in the OPAC Procedure Name attribute.

Properties Dialog

The Properties dialog for the block is shown below:



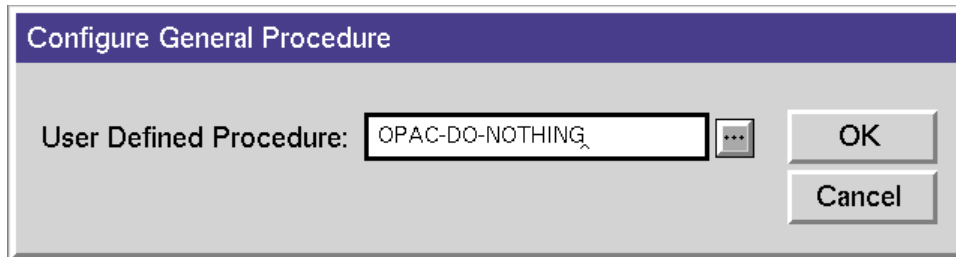
Attributes

Attribute	Data Type	Default Value	Possible Values
Names	symbol	none	any valid symbol
Description	text	""	any text

General Procedure

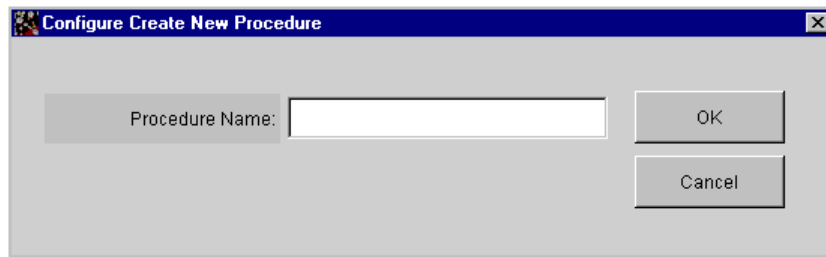


Use the General Procedure block to embed your own G2 actions within an OPAC procedure. To configure the General Procedure block, select Configure from the right-click drop-down menu. The Configure dialog box appears:



Specify the name of your G2 procedure in the User Defined Procedure attribute. The default procedure name is OPAC-Do-Nothing. If you have already defined the procedure, click on the OK button.

If you have not yet defined the G2 procedure, you can use this block to create the procedure. To create the procedure, click on the ellipsis button next to the User Defined Procedure attribute. The following dialog box appears:



Enter the name for the new procedure in the Procedure Name attribute and click on OK. This action creates a Procedure block and displays the new block on the same workspace with the General Procedure block. To set the properties of this Procedure block and edit the text of the procedure, see the section “Procedure Block” in this chapter.

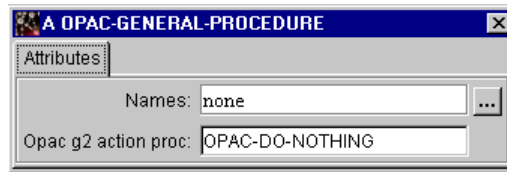
The only argument passed to a procedure called by the block is the token. Error handling is automatic; the error handler automatically returns the error-name (symbol) and error-text (text) of any error that occurs.

Attributes

Attribute	Data Type	Default Value	Possible Values
opac-g2-action-proc	symbol	opac-do-nothing	any G2 procedure name

Properties Dialog

The Properties dialog for the block is shown below:



Send SMH Message



The Send SMH Message block creates and sends a message to the specified message server. The block sends messages only to servers in the same G2 process.

The attributes of the block match the calling arguments for the SMH Create Message block, the procedure API for creating and sending any message. The calling arguments needed for the SMH are specified as attributes of this block.

For information on message handling refer to the *Integrity User's Guide*.

Configure Send SMH Message

Message Text:

Additional Text:

◆ Server:

◆ Target:

◆ Sender:

◆ Category:

◆ Window:

◆ Priority:

Display Options: append ignore duplicate replace

acknowledgement timestamp history

Lifetime: weeks: days:

hours: minutes: seconds:

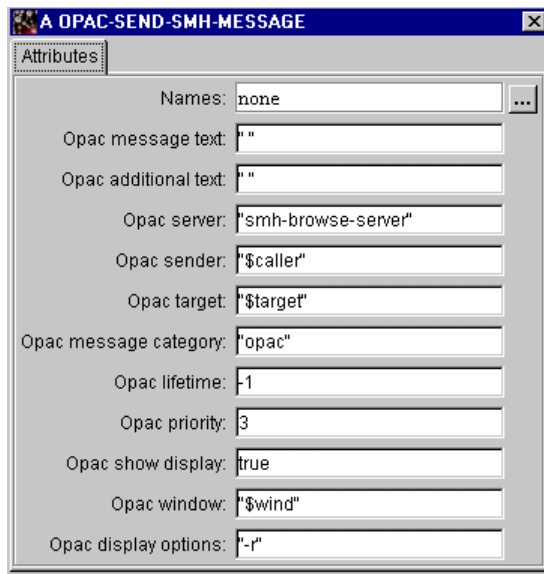
Attributes

Attribute	Data Type	Default Value	Possible Values
opac-message-text	text	""	any text of message. If the value contains (\$) references, they are resolved.
opac-additional-text	text	""	any text. If the value contains (\$) references, they are resolved.
opac-server	text	"smh-browser-server"	any valid message server
opac-sender	text	"\$caller"	any object
opac-target	text	"\$target"	any subclass of opfo-managed-object or opfo-containment-object
opac-message-category	text	"opac"	any user-defined category. If the value contains (\$) references, they are resolved.
opac-lifetime	number	-1	lifetime of message in seconds. Any number < 0 means infinite lifetime
opac-priority	number	3	any priority. Local integer parameters can also be used as they are resolved.
opac-show-display	symbol	true	true or false

Attribute	Data Type	Default Value	Possible Values
opac-window	text	"\$wind"	a g2-window or object
opac-display-options	text	"-r"	"-nolist" and/or "-nack" and/or "-i" or "-a" or "-r"

Properties Dialog

The Properties dialog for the block is shown below:



Historical Message Query



The Historical Message Query block specifies a query against the message history. The block returns the result to the stack. You can use the following criteria in a query:

- **Target** - The target of the message. Substitution variables such as `$stack` or `$target` are allowed. Wildcards such as `(*)` or `(?)` are not allowed.
- **Sender** - The sender of the message. This can be a reference to a single item, or may be left blank to indicate that any sender is an acceptable match. Substitution variables like `$stack` or `$sender` are allowed.
- **Message Category Pattern** - The category of the messages. Wildcard characters `"*"` and `"?"` are allowed in the category specification. This attribute does not allow local parameter substitution.
- **Begin Pattern Match with Character** - The character position to begin the match at within the Messages Category Pattern attribute.
- **Query Length** - The length of period for the query, in minutes. For instance, with Query Length set to 5, the query would count the number of messages meeting the search criteria within the last 5 minutes.

The screenshot shows the 'Configure Historical Message Query' dialog box. It features a title bar and several input fields. The 'Target' field contains '\$target', the 'Sender' field contains '\$caller', and the 'Message Category Pattern' field contains 'opac'. To the right of these fields is a list box containing '\$stack' and '\$caller'. Below these fields are two spinners: 'Begin Pattern Match with Character' set to '1' and 'Query Length (minutes)' set to '5'. At the bottom are 'OK' and 'Cancel' buttons.

This block is usually followed by a Comparison Decision block, which can test the number returned from the history query against a specified number. For details, see [Comparison Decision](#).

Attributes

Attribute	Data Type	Default Value	Possible Values
opac-match-time-interval-minutes	number	1.0e6	any number representing minutes or any local parameter representing a number.
opac-target	text	“\$target”	any subclass of opfo-managed-object or opfo-containment-object, any opac-local-item, an attribute of some object, or the ext-name of a managed or containment object. If \$stack is used, the stack is not consumed.
opac-sender	text	“”	any, object, any opac-local-item, an attribute of some object, or the ext-name of a managed or containment object. If \$stack is used, the stack is not consumed.
opac-message-category	text	“”	any user-defined category.
opac-message-category-starting-position	number	1	any number valid in the range of the length of the category of message

Properties Dialog

The Properties dialog for the block is shown below:

A screenshot of a software dialog box titled "A OPAC-HISTORICAL-MESSAGE-QUERY". The dialog has a standard Windows-style title bar with a close button. Below the title bar is a tab labeled "Attributes". The main area of the dialog contains several text input fields:

- Names: none
- Opac match time interval minutes: 1000000.0
- Opac target: "\$target"
- Opac sender: ""
- Opac message category: ""
- Opac message category starting position: 1

Hide Workspace

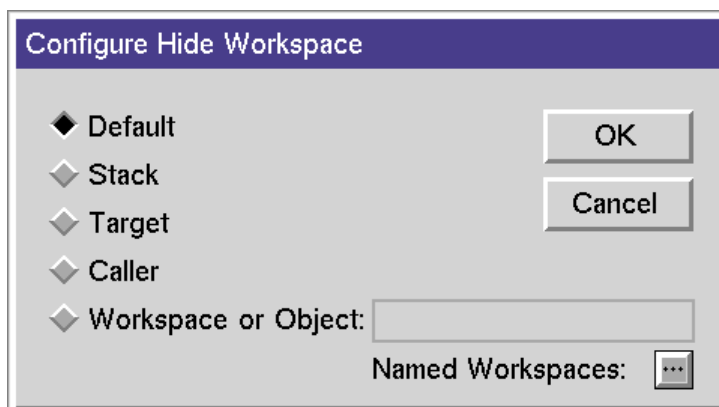


The Hide Workspace block hides the workspace specified in the Workspace Spec attribute of the block.

The Workspace Spec attributes of this block are:

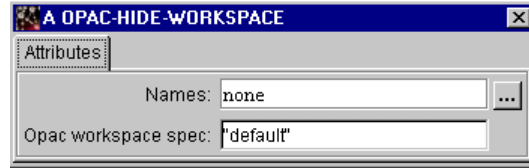
- Default - If the Hide Workspace block has a subworkspace, selecting this option hides the subworkspace. If the block has no subworkspace, selecting this option hides the workspace that contains the block.
- Stack - If the first item in the general stack of the token is a workspace, selecting this option hides that workspace and removes it from the stack.
- Target - If the target is a workspace that has a subworkspace, selecting this option hides the subworkspace. If the target is a workspace without a subworkspace, the option hides the workspace. If the target is an object other than a workspace, the option hides the workspace containing the target object.
- Caller - If the caller is a workspace that has a subworkspace, selecting this option hides the subworkspace. If the caller is a workspace without a subworkspace, the option hides the workspace. If the caller is an object other than a workspace, the option hides the workspace containing the caller object.
- Workspace or Object - Select this option to specify a symbolic name of a workspace or object. If you specify the name of a workspace that has a subworkspace, this option hides the subworkspace. If the specified workspace does not have a subworkspace, the option hides the workspace. If you specify any other item, the option hides the item.

The windows hidden are based on the window set for caller of the token. For instance, if the caller is a single G2 window, the workspace is hidden only for that window.



Properties Dialog

The Properties dialog for the block is shown below:



Show Workspace Not Stacked



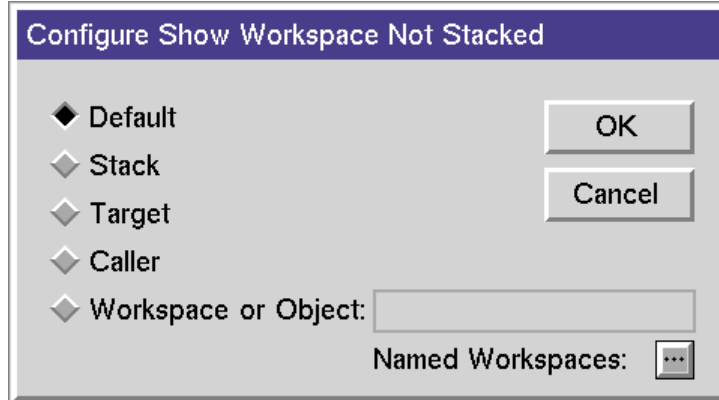
Two variations exist for the Show Workspace block. Both blocks display the workspace you specify:

- The Show Workspace block shows the specified workspace and places the workspace on the general stack of the token.
- The Show Workspace Not Stacked block shows the specified workspace, but does not place the workspace on the general stack of the token.

Possible Values

Specify the workspace to be displayed in the Workspace Spec attribute of the block. The choices are:

- Default - If the Show Workspace block has a subworkspace, selecting this option shows the subworkspace. If the block does not have a subworkspace, select this option to show the workspace that contains the block.
- Stack - If the first item in the general stack of the token is a workspace, selecting this option shows the workspace and removes it from the stack.
- Target - If the target is a workspace that has a subworkspace, selecting this option shows the subworkspace. If the target is a workspace without a subworkspace, the option shows the workspace. If the target is an object other than a workspace, the option shows the workspace containing the target object.
- Caller - If the caller is a workspace that has a subworkspace, this option shows the subworkspace. If the caller is a workspace without a subworkspace, the option shows the workspace. If the caller is an object other than a workspace, the option shows the workspace containing the caller object.
- Workspace or Object - Select this option to specify a symbolic name of a workspace or object. If you specify a workspace that has a subworkspace, this option shows the subworkspace. If the specified workspace does not have a subworkspace, the option shows the workspace. If you specify any other item, the option shows the item.

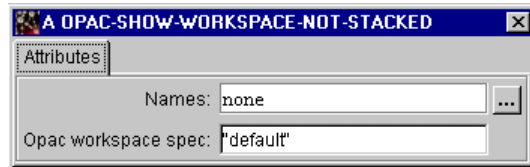


Attributes

Attribute	Data Type	Default Value	Possible Values
opac-workspace-spec	symbol	default	See possible values for Properties Dialog .

Properties Dialog

The Properties dialog for the block is shown below:



Show Workspace



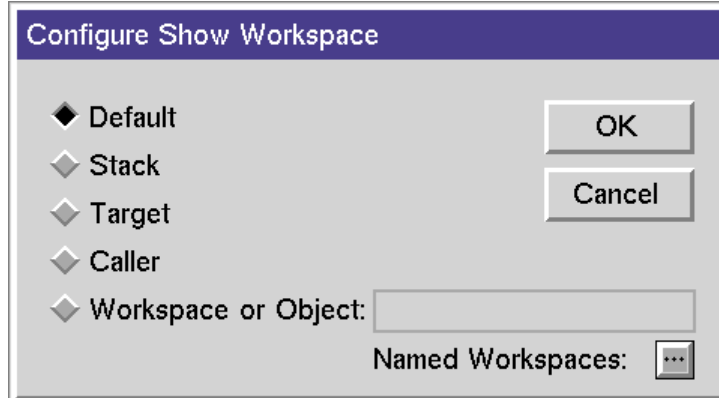
Two variations exist for the Show Workspace block. Both blocks display the workspace you specify:

- The Show Workspace block shows the specified workspace and places the workspace on the general stack of the token.
- The Show Workspace Not Stacked block shows the specified workspace, but does not place the workspace on the general stack of the token.

Possible Values

Specify the workspace to be displayed in the Workspace Spec attribute of the block. The choices are:

- Default - If the Show Workspace block has a subworkspace, selecting this option shows the subworkspace. If the block does not have a subworkspace, select this option to show the workspace that contains the block.
- Stack - If the first item in the general stack of the token is a workspace, selecting this option shows the workspace and removes it from the stack.
- Target - If the target is a workspace that has a subworkspace, selecting this option shows the subworkspace. If the target is a workspace without a subworkspace, the option shows the workspace. If the target is an object other than a workspace, the option shows the workspace containing the target object.
- Caller - If the caller is a workspace that has a subworkspace, this option shows the subworkspace. If the caller is a workspace without a subworkspace, the option shows the workspace. If the caller is an object other than a workspace, the option shows the workspace containing the caller object.
- Workspace or Object - Select this option to specify a symbolic name of a workspace or object. If you specify a workspace that has a subworkspace, this option shows the subworkspace. If the specified workspace does not have a subworkspace, the option shows the workspace. If you specify any other item, the option shows the item.

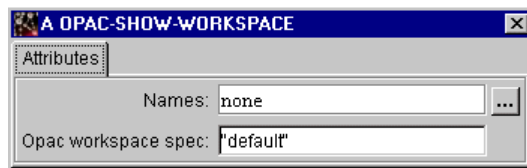


Attributes

Attribute	Data Type	Default Value	Possible Values
opac-workspace-spec	symbol	default	One of the following: <ul style="list-style-type: none"> • default • stack • target • caller • workspace or object symbolic name

Properties Dialog

The Properties dialog for the block is shown below:



Block Pause Capability



Placing an Block Pause Capability block on a workspace displays OPAC tokens and their movement through the processes on that workspace. The Block Pause Capability block does not have to be connected to any other block for this action to occur.

If you attach the Block Pause Capability block to another OPAC block, it pauses execution of the OPAC procedure at that point, presents a dialog of choices to the operator, and awaits operator approval to continue with processing the next step.

The token stops at the block preceding the one with the block pause icon. When this block executes, OPAC generates an operator dialog. The dialog offers a choice of continuing or aborting the procedure.

This block also has a Timeout attribute. At the end of the specified timeout, processing continues. Specify a timeout when you require operation to continue if an operator is not present.

Specify header text for the dialog box in the Header Text attribute. Specify the text choice for continuing the procedure in the Continue Text attribute. You can use text substitution variables, such as `$target`, in either text attribute.

Configure Block Pause Capability

Header Text: Pause in task \$task, for target=\$target. Select choice;

Continue Text: Continue with next step, \$bloc ...

Timeout: weeks: 0 days: 0 hours: 0 minutes: 1 seconds: 0

OK Cancel

Attributes

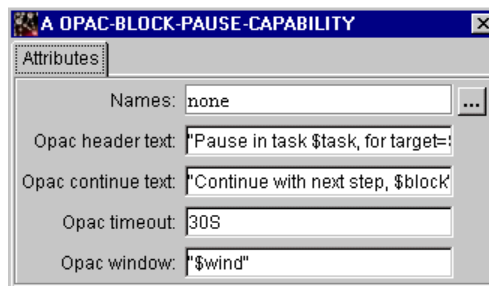
Attribute	Data Type	Default Value	Possible Values
opac-header-text	text	“Pause in the \$task, for target = \$target. Select choice:”	Any text message. Will resolve local names beginning with \$’s.
opac-continue-text	text	“Continue with next step, \$block”	Any text message. Will resolve local names beginning with \$’s.
opac-timeout	symbol	1m	Any number followed by one of: s = seconds m = minutes h = hours d = days w = weeks

This block is commonly used for debugging, as well as for obtaining operator approval of actions. It is also used for demos in order to pace actions taken.

Note Displaying token movement is CPU-processing-intensive; processing time is increased by approximately a factor of ten.

Properties Dialog

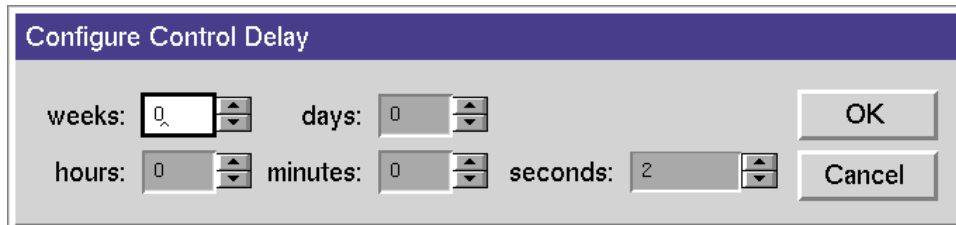
The Properties dialog for the block is shown below:



Control Delay



The Control Delay block causes a timed delay, specified by the Delay attribute. The default time unit is seconds. The time units can be specified by appending (s) for seconds, (m) for minutes, (h) for hours, (d) for days, and (w) for weeks.

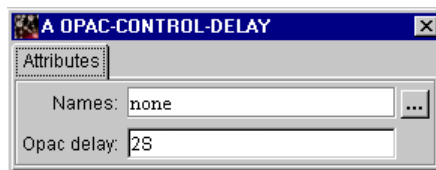


Attributes

Attribute	Data Type	Default Value	Possible Values
opac-delay	symbol	2s	Any number followed by one of: s = seconds m = minutes h = hours d = days w = weeks

Properties Dialog

The Properties dialog for the block is shown below:



Task Kill



The Task Kill block aborts processing and deletes the token. The General Stack transient items in the general stack is deleted. This action has no effect on any other tokens that may be processing.

Note This is a nonconfigurable block.

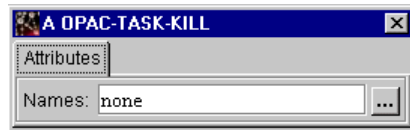
Attributes

Attribute	Data Type	Default Value	Possible Values
-----------	-----------	---------------	-----------------

N/A

Properties Dialog

The Properties dialog for the block is shown below:



Task Spawn



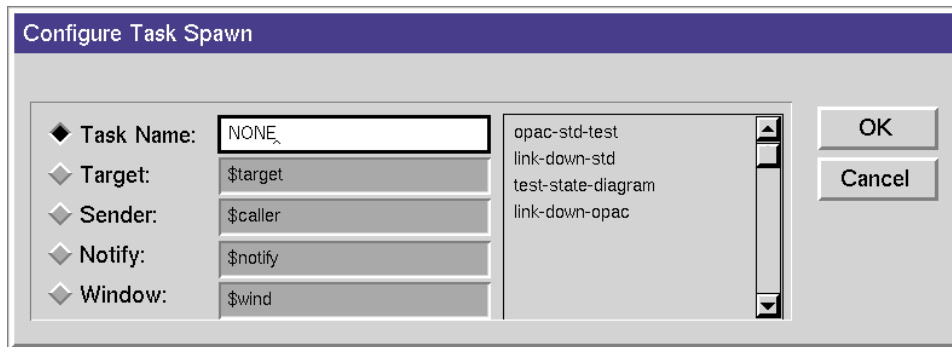
The Task Spawn block starts another OPAC procedure from within an OPAC procedure. A new token is created, and it now has a life independent of the originating token, with no return of control to the originating token. Both tokens continue processing OPAC blocks.

No further communication occurs between the two tokens. The originating token can be deleted with no effect on the new token. This is analogous to a spawn in operating systems, such as UNIX, that is launched with *nohup*. It is equivalent to a start of a process in the G2 procedural language. If you want the graphical equivalent of a G2 call of a procedure, however, you should use a Subtask block instead of a Task Spawn block.

Set the Task Name attribute to the name of the Subtask Start block that begins the new procedure.

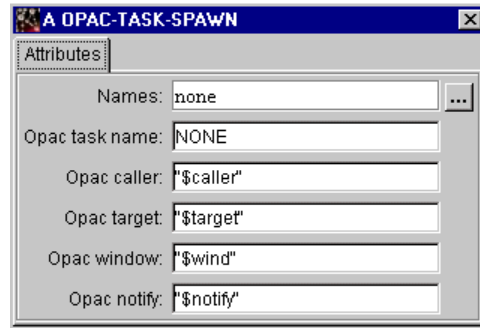
Arguments are passed in a manner similar to the Subtask block.

The context **\$caller**, **\$target**, **\$window** and **\$notify** are specified in the attributes of the Task Spawn block, with attributes named, respectively: **caller**, **target**, **window**, and **notify**. The default values are set, respectively, to **\$caller**, **\$target**, **\$wind**, **\$notify**. The newly-spawned task, **\$task** now becomes the name of the Subtask Start block. However, local parameters in the calling procedure are not available to blocks in the spawned task.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-task-name	symbol	no name	Name of any opac-subtask-start block
opac-caller	text	“\$caller”	Any object
opac-target	text	“\$target”	Any subclass of opfo-managed object or opfo-containment-object
opac-window	text	“\$wind”	A G2-window or object
opac-notify	text	“\$notify”	Any message server

Subtask



The Subtask block is the equivalent of a subroutine or function call in other languages in that it transfers control and expects a return of control. The Name attribute of the Subtask block must be the same as the Subtask Name attribute of an Subtask Start block.

OPAC allows recursive calls to a subtask. The task can also call itself. You must ensure that there is a way to return, ultimately, from the task.

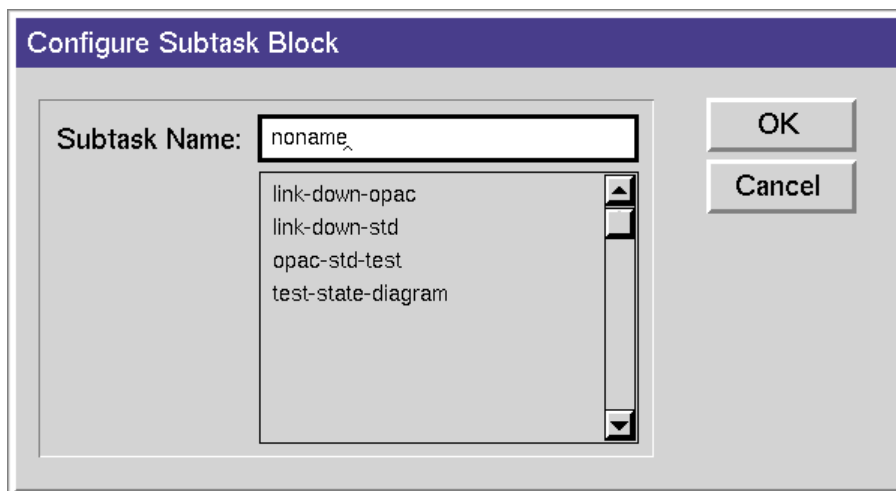
The standard context is preserved during a subtask call. That is, `$target`, `$caller`, `$window`, and `$notify` are preserved for use in all subtasks. In the subtask, `$task` now becomes the name of the subtask. `$task` returns to its previous value when the subtask returns.

User-defined local parameters in a procedure are not available in the subtasks unless you pass them to the subtask. Unless passed, they are local only to the procedure in which they are defined.

You can pass items to a subtask by placing the items on the General Stack. However, you will also pass arguments, as described in [Passing Arguments to a Subtask](#).

Note Most of the information needed in any procedure is often contained in the objects referenced by `$target`, `$sender`, which are already available without passing arguments.

The ability or necessity to pass arguments to the subtask is a major differentiator between Subtask blocks and Macro blocks. For a Macro block, no arguments are passed.

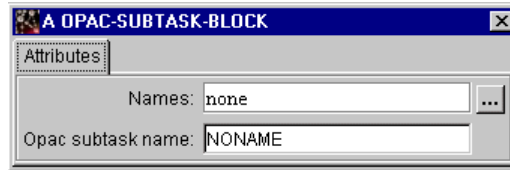


Attributes

Attribute	Data Type	Default Value	Possible Values
opac-subtask-name	symbol	no name	Name of any opac-subtask-start block

Property Dialog

The Properties dialog for the block is shown below:



Macro



The Macro block calls a macro, transferring control to the macro until the macro procedure is complete. Arguments cannot be passed to the macro.

Specify the name of the macro with the Macro Name attribute. This Macro Name must be the actual G2 name of an Subtask Start block.

Control is returned to this calling Subtask block when a Subtask Completion block is encountered (see [Subtask Completion](#)).

A macro allows recursive calls (i.e., the macro can call itself.) However, be sure that you always provide a way to return from the macro.

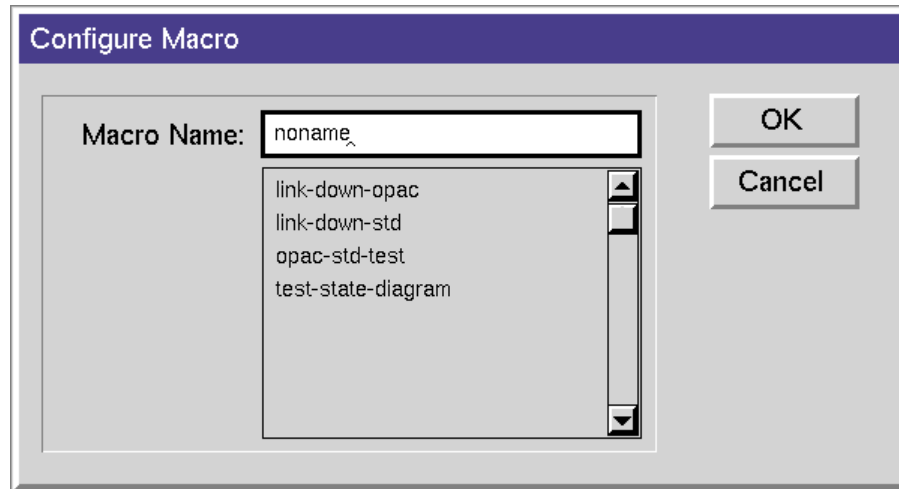
Caution A stack of calls is kept internally, and, without returns, this stack can grow to fill all available virtual memory.

The standard context of `$target`, `$caller`, `$window`, and `$notify` are preserved during a macro call. The macro `$task` now becomes the name of the macro; however, `$task` returns to its previous value when the macro returns. Unlike a subtask, though, the local parameters in the calling procedure are also available to blocks in the macro.

No arguments can be passed to a macro, and no local parameters are expected in the macro; they are ignored. The complete calling context remains unchanged, including `$target`. The existing local parameters from the calling procedure are also still in effect. Thus, in a macro, you can refer to local parameters in the calling program, unlike the case of a subtask, where the local parameters are only known within that subtask.

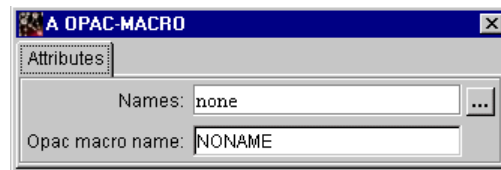
A macro accomplishes a result in some ways similar to using a Connection-Post to transfer control to a procedure on another workspace. However the connection post method does not provide a certain return of control to the calling procedure. If multiple procedures passed to the procedure through a connection post, because the called procedure does not track which procedure was the caller.

Because of the resemblance to passing control through connection posts, the icon for a Macro block includes the suggestion of two Connection Post icons.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-macro-name	symbol	no name	Name of any opac-subtask-start block

Subtask Completion



The Subtask Completion block indicates the end of an OPAC procedure.

When an OPAC token encounters an Subtask Completion block, it returns to any existing calling task. If no calling task exists, it deletes the token and any associated storage. It also deletes any space allocated to local variables as well, unless it is called from an Macro block.

More than one Subtask Completion block is allowable in a graphical program Subtask block.

Note The Subtask Completion block is a non-configurable block.

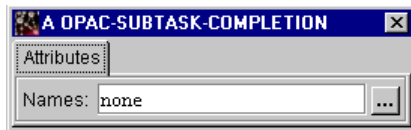
Attributes

Attribute	Data Type	Default Value	Possible Values
-----------	-----------	---------------	-----------------

N/A

Properties Dialog

The Properties dialog for the block is shown below:



Subtask Start



The Subtask Start block defines the beginning of an OPAC procedure. The Subtask Name attribute of this block identifies the procedure. The block can be called by a Subtask block, a G2 procedure, or it can be launched manually.

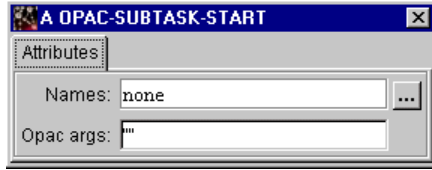
The Subtask Start block has an Argument attribute to specify values or items passed in as arguments to a subtask.

Attributes

Attribute	Data Type	Default Value	Possible Values
opac-subtask-name	symbol	no name	name of any opac-subtask-start block
opac-args	text	""	comma separated local names

Properties Dialog

The Properties dialog for the block is shown below:



Procedure Statement



The Procedure Statement block allows users to create G2 procedures within the OPAC block language. Legal G2 procedure statements, actions, and for loops are allowed.

The following rules apply when creating procedure statements, using the Procedure Statement block.

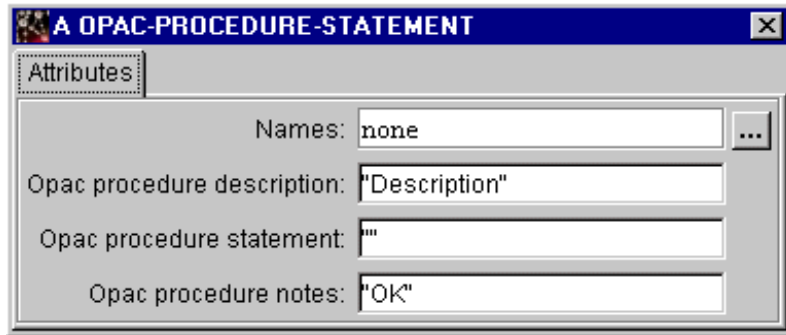
Note References are case insensitive.

- Procedures understand \$TARGET, \$TOK, \$WIND, \$CALLER, \$NOTIFY, \$STACK, and \$BLOCK references although these must be used in the proper context.
- Normal OPAC substitution variables (references to local parameters) are allowed. Substitution variables must be preceded by the (\$) reference, such as \$local-name.
- Unknown substitution variables (indicated by starting with \$) are assumed to be local names within the procedure and a local name of type *item or value* is created within the procedure.
- References to (\$) symbols (substitution variables) are pointers to the actual item.

The screenshot shows a dialog box titled "Configure Procedure Statement". It contains three input fields: "Description" (containing "Description"), "Status" (containing "Inactive"), and "Statement" (which is empty). To the right of these fields are two buttons: "OK" and "Cancel".

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-procedure description	text	"Description"	any text
opac-procedure-statement	text	""	any valid procedure statement(s). Local names beginning with \$'s are resolved.
opac-procedure-notes	text	"Inactive"	Used as a status only. Does not operate on this attribute.

The OPAC Procedure block is driven by a **whenever** rule that executes any time the procedure statement is modified.

- A subworkspace is created for the block that serves as a container for the procedure.
- A clone of a predefined procedure skeleton is created. This procedure has built in local bindings for *standard* references to items such as \$stack, \$token.
- The procedure statement is scanned to find all (\$) references. The (\$) is stripped from the text to be inserted into the procedure. The resulting symbol matches with local bindings.

- References to things other than the *standard* symbols cause the creation of a local name within the procedure. Local name declarations are assigned by an “if then else statement” that looks for an OPAC local variable connected to the Start block being passed by the token; if one exists, a binding to that item is created, otherwise the local name is assigned a value of *none*
- The name of the procedure is defined as `opac-statement-procedure-[random(1,10000)]`.
- The text of the skeleton is changed, transferred to the subworkspace, and made permanent.
- Errors and inconsistencies are displayed in the Procedure Notes attribute of the block.

Referencing the Stack

You must use the proper syntax for inserting and removing items from the stack. Some simple examples are:

{Pop top Item from the stack}

remove the first item from \$STACKPTR;

{Delete the top item from the stack;}

delete \$STACK;

{Insert an item into the stack;}

insert \$test at the beginning of \$stackptr;

{change the value of the top value on the stack}

conclude that \$STACK = 5;

{assign the value of the top value on the stack to a local parameter test}

\$TEST = \$stack;

Examples

The following examples are for procedure statements and functions:

Increment the priority of \$target by local-test.

conclude that the `_opfo-highest-message-priority` of \$target = the `_opfo-highest-message-priority` of \$target + \$local-test;

List all the procedure blocks on this workspace.

for \$statement = each `opac-procedure-statement` upon the workspace of \$block
do

inform the operator on the workspace of \$block for the next 20 seconds that

```
    “[ the opac-procedure-description of $statement]”;  
end;
```

Create a list of items connected to the test object and insert the list onto the stack.

```
create an item-list $lst;  
for $obj = each object connected to $target do  
    insert $obj at the end of the item-list list $lst;  
end
```

Show the list without comments.

```
transfer $lst to the workspace of $BLOCK at (the item-x-position of $block - 75,  
    the item-y-position of $block);  
insert $lst at the beginning of $stackptr;  
    {Pop and delete the list just put on the stack}  
delete $STACK;
```


Procedure Template



The Procedure Template block provides a template for you to define your own user block definitions. The Procedure Template block is a G2 feature designed to help you build and define a procedure by providing a template for a General Procedure block or a Set Local Parameter block. Configure the block through the Properties dialog G2 attribute table or by selecting the Edit option of the procedure menu.

You can define your own G2 procedures and specify the procedure name in the G2 Action Procedure of the blocks. This template displays a sample procedure to build from and supplies the additional information on OPAC Tokens.

Note The OPAC procedure template does not show opac-token-error-handler usage information.

An OPAC G2 procedure must return error-name and error-text at the end of the procedure and use an on-error statement to return these values.

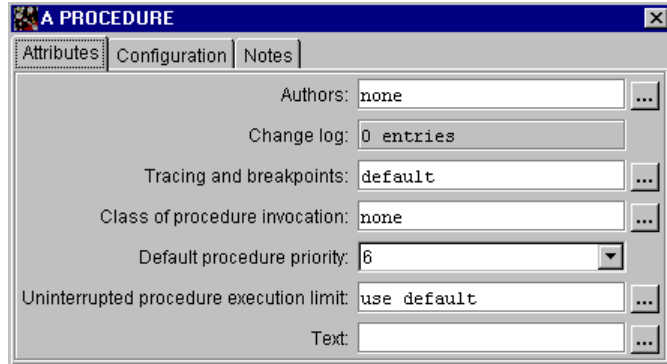
The following is an example of a G2 procedure:



a procedure	
Notes	INCOMPLETE
Authors	none
Item configuration	none
Tracing and breakpoints	default
Class of procedure invocation	none
Default procedure priority	8
Uninterrupted procedure execution limit	use default
<pre> OPAC-STATEMENT-PROCEDURE-TEMPLATE(_TOK: class opac-token) = (symbol,text) error-name: symbol = the symbol OK; error-text text = " "; _BLOCK: class opac-syntax-element = the opac-syntax- element that is the opac-block-of _TOK; _TARGET: class item = the item that is an opac-target- of _TOK; _WIND: class item = the item that is an opac-window-of _TOK; _CALLER: class item = the item that is the opac-caller- of _TOK; _NOTIFY: class item = the item that is opac-notified-by _TOK; LVL: class g2-list = the first g2-list in the _opac-local- variable-list of _TOK; ITM: class item; {*****} {Example of getting a local-parameter from the _opac- local-variable-list of the token} _x item-or-value = (if there exists a parameter ITM in LVL such that (the opac-local-name of ITM = the symbol x) then ITM else the symbol no-value); {*****} begin {#####} {Your code goes here} {#####} return error-name, error-text; end on error (error-name, error-text) return error-name , error-text ; end </pre>	

Properties Dialog

The Properties dialog for the block is shown below:



Connection Post



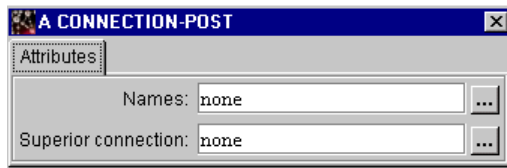
The Connection Post block allows you to connect OPAC procedures across workspaces. The block is not a configurable OPAC block; it is a standard feature of Gensym's G2 product that is also used by OPAC. Use these blocks to maintain connectivity between workspaces by defining two connection ports with the same G2 name.

For example, if a procedure requires iteration over all objects connected to a specified object, the Connection Post block allows the iteration to occur on objects located on multiple workspaces .

Note The Connection Post block does not have a configuration dialog box. Edit the Properties dialog box of the block to configure the block.

Properties Dialog

The Properties dialog for the block is shown below:



Decisions Palette

Describes the blocks in the Decisions palette.

Introduction	58
Comparison Decision	59
2-Way Decision	62
2-Way Manual Decision	66
3-Way Manual Decision	69
4-Way Manual Decision	71
2-Way Pattern Decision	73
3-Way Pattern Decision	76
4-Way Pattern Decision	78
2-Way Pattern Decision By Symbol	81



Introduction

Decisions blocks are used in conjunction with General Action blocks for decision actions. Manual, Pattern Match and Consume Decision From Stack blocks are available in either two, three, or four decision choices. The Use Entered Decision Procedure block is a user defined block.

Here is the Decisions palette:



Decisions palette blocks can only accept connections to their stubs. No external connections are allowed such as connections from a Block Pause Capability block.

Comparison Decision



The Comparison Decision block provides two-way branching capability. Branching is based on evaluation of a relational expression that evaluates to either True or False.

When the token is input to the block, the expression is evaluated. If the expression evaluates to True, the token is routed through output 1. If the expression evaluates to False, the token is routed through output 2.

Enter the expression in the Comparison attribute of the configuration dialog.

Specify the attributes as follows:

- Comparison - A relational expression, which consists of two arithmetic expressions separated by a relational operator.
- Arithmetic Expression - Relational operators to be used in the comparison relational expression.
- Choice If Error in Comparison - Default expression for errors.
- Choice 1 - The choice to be taken if True.
- Choice 2 - The choice to be taken if False.

Allowable relational operators are:

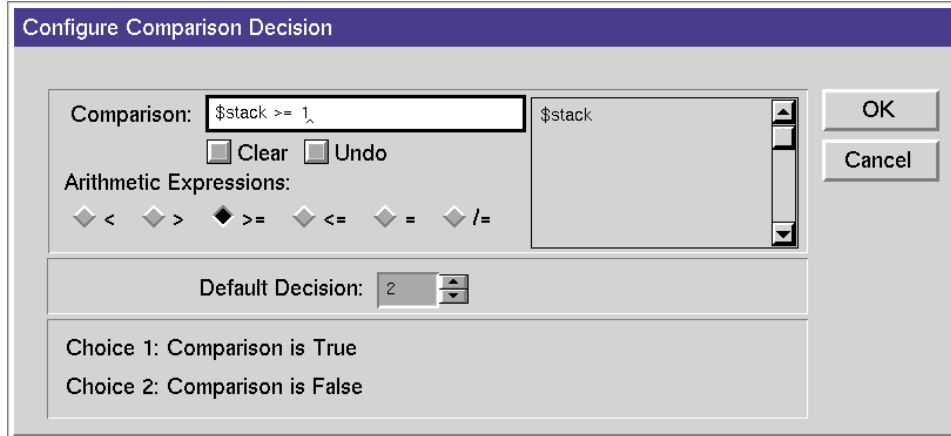
<
 <=
 >
 >=
 =
 /=

Examples:

`$stack >= 1`

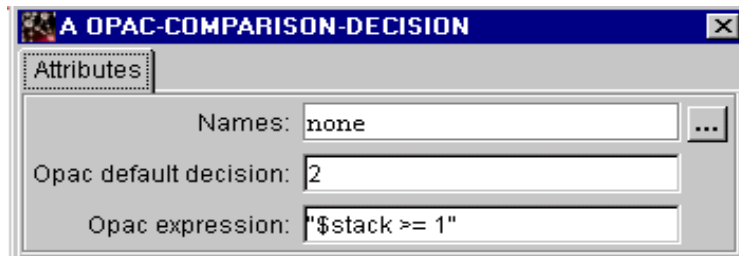
`the _opfo-highest-message-priority of $msg > the _opfo-highest-message-priority of $target`

Allowable arithmetic expressions on the left-hand side and right-hand side are parameter names, or names with attributes. Substitution parameters may be used, the substitution is done before evaluation. Standard references may be made, such as `$stack`, `$target`, references to local parameters.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-expression	text	“\$stack >= 1”	Any expression. (\$) references will be resolved. Cannot use any functions in the expression. Must follow sample format. Possible relational operators include: >=, <=, >, <, =, and /=.
opac-default-decision	number	2	The decision (1 or 2) to be taken in case of a timeout or error condition.

2-Way Decision



The 2-Way Decision block provides branching capability based on a user-defined G2 procedure.

For example, the block shown below has been specified to use the procedure Consume Decision From Stack. In this example, the block uses the top item on the stack to decide which path of the block to take. If the value of the item on the stack is less than or greater than the number of output paths for the block, the default decision is taken.

The dialog box is titled "Configure User Defined 2 Way Decision". It contains the following fields and controls:

- User Defined Decision Procedure:** A text field containing "OPAC-CONSUME-DECISION-FROM-STACK" with an "OK" button to its right.
- Choice 1 Description:** An empty text field.
- Choice 2 Description:** An empty text field.
- Timeout:** A checked checkbox followed by a group of spinners for "weeks: 0", "days: 0", "hours: 0", "minutes: 0", and "seconds: 0".
- Choice after timeout or error:** A spinner set to "2".
- Buttons:** "OK" and "Cancel" buttons are located on the right side of the dialog.

Properties Dialog

The Properties dialog for the block is shown below:

The dialog box is titled "A OPAC-2-WAY-DECISION" and has a tab labeled "Attributes". It contains the following fields:

- Names:** A text field containing "none" with a browse button "...".
- Opac decision proc:** A text field containing "OPAC-DEFAULT-DECISION-PR".
- Opac timeout:** A text field containing "-1".
- Opac default decision:** A text field containing "2".
- Opac source:** A text field containing "".
- Opac choice 1 description:** A text field containing "".
- Opac choice 2 description:** A text field containing "".

Attributes

Attributes	Data Type	Default Value	Possible Values
opac-decision-proc	symbol	opac-consume-decision-from-stack	User-defined G2 procedure with arguments.
opac-choice-1-description	text	""	Any text; (\$) references will be resolved
opac-choice-2-description	text	""	Any text; (\$) references will be resolved
opac-timeout	number	-1	Any number. If a default decision should be taken after a certain time period, specify it here, using appropriate time specifications.
opac-default-decision	number	2	The decision (1 or 2) to be taken in case of a timeout or error condition.

Special Instance: If Token Error Free Block

The If Token Error Free block is an instance of the standard 2-Way Decision block. This instance checks for errors in the token. To create this instance, use the standard block and specify the If Token Error Free decision procedure in the User Defined Decision Procedure attribute.

A token that is input to the block is output through one of the two output options. If the token is error-free, the token continues through output path 1; otherwise, it is output through path 2.

Usage

One common use of this block is in conjunction with a following manual decision block that allows operator intervention. If the token is in error, the operator sees an indicative message on the smh-error-server message server and can take appropriate action.

This block is important in many applications in which calls to external systems may hang up or time out.

In the event of an error, blocks usually pass on the token, despite the error, so that information about events is not lost. This block is useful in the process flow before major actions are taken in the event of an error.

Attributes

Attribute	Data Type	Default Value	Possible Values
opac-decision-proc	symbol	opac-if-token-error-free	Name of the user-defined decision procedure.
choice 1 description	text	No error	not used
choice 2 description	text	Token in error	not used

Attribute	Data Type	Default Value	Possible Values
opac-timeout	number	-1	Any number. If a default decision should be taken after a certain time period, specify it here, using appropriate time specifications.
opac-default decision	number	2	The decision (1 or 2) to be taken in case of a timeout or error condition.

2-Way Manual Decision



The 2-Way Manual Decision block provides a two-way branching capability for OPAC, similar to the 2-Way Decision block. The 2-Way Manual Decision block differs, however, by presenting a prompt to the operator who can manually enter a response. The manual response of the operator determines the branch through which processing continues and the token is output.

The prompt is presented when the token is input to the block. Specify the text presented to the operator that describe the choices by entering the text in the Manual Choice 1 Description and Manual Choice 2 Description attribute.

The dialog presented to the operator is generated automatically, based on the choice descriptions, and the header text is specified by the Header Text for Manual Choices attribute. Substitution variables are allowable in these attributes.

The optional timeout attribute allows you to specify:

- a time interval for which to await operator response
- the branch to choose if operator response does not occur within the specified time interval.

Configure 2 Way Manual Decision

Header Text for Manual Choices: In task \$task, for target= \$target, select choice; OK

Manual Choice 1 Description:

Manual Choice 2 Description:

Timeout:

weeks: 0 days: 0

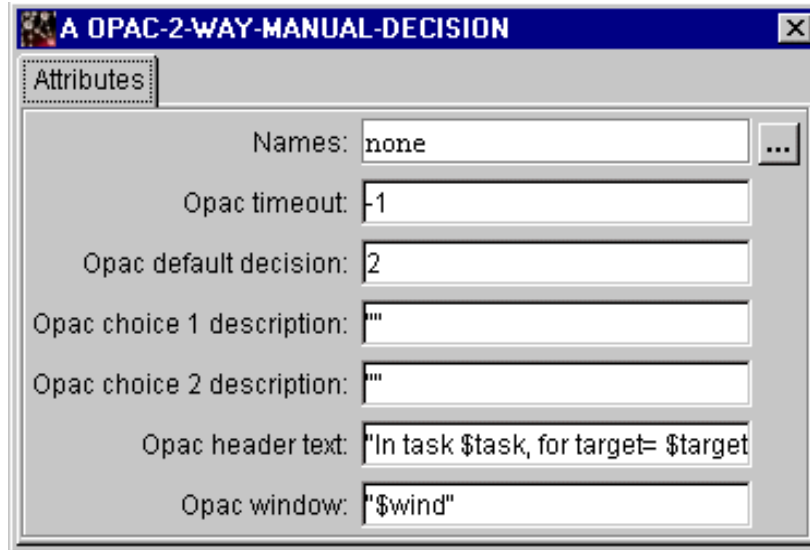
hours: 0 minutes: 0 seconds: 0

Choice after timeout: 2

Cancel

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-header-text	text	"In task \$task, for target = \$target, select choice:"	Any text; (\$) references will be resolved
opac-choice-1-description	text	""	Any text; (\$) references will be resolved
opac-choice-2-description	text	""	Any text; (\$) references will be resolved

Attribute	Data Type	Default Value	Possible Values
opac-timeout	number	-1	Any number. If a default decision should be taken after a certain time period, specify it here, using appropriate time specifications.
opac-default-decision	number	2	The decision (1 or 2) to be taken in case of a timeout or error condition.

3-Way Manual Decision



The 3-Way Manual Decision block provides three-way branching capability, prompting an operator to decide to continue processing through one of three branches. The block allows specification of a timeout default decision branch. Operation is similar to the 2-Way Manual Decision block, except three choices are presented to the operator.

Properties Dialog

The Properties dialog for the block is shown below:

Attributes

Attribute	Data Type	Default Value	Possible Values
opac-header-text	text	“In task \$task, for target = \$target, select choice:”	Any text; (\$) references will be resolved
opac-choice-1-description	text	““	Any text; (\$) references will be resolved
opac-choice-2-description	text	““	Any text; (\$) references will be resolved
opac-choice-3-description	text	““	Any text; (\$) references will be resolved
opac-timeout	number	-1	Any number. If a default decision should be taken after a certain time period, specify it here, using appropriate time specifications:
opac-default-decision	number	3	The decision (1, 2, or 3) to be taken in case of a timeout or error condition.

4-Way Manual Decision



The 4-Way Manual Decision block provides four-way branching capability, prompting an operator to decide to continue processing through one of four branches. The block allows specification of a timeout default decision branch. Operation is similar to the 2-Way Manual Decision block, except four choices are presented to the operator.

Configure 4 Way Manual Decision

Header Text for Manual Choices: In task \$task, for target= \$target, select choice. OK

Manual Choice 1 Description: [] Cancel

Manual Choice 2 Description: []

Manual Choice 3 Description: []

Manual Choice 4 Description: []

Timeout:

weeks: 0 days: 0

hours: 0 minutes: 0 seconds: 0

Choice after timeout: 4

Properties Dialog

The Properties dialog for the block is shown below:

A OPAC-4-WAY-MANUAL-DECISION

Attributes:

Names: none ...

Opac timeout: -1

Opac default decision: 4

Opac choice 1 description: ""

Opac choice 2 description: ""

Opac choice 3 description: ""

Opac choice 4 description: ""

Opac header text: "In task \$task, for target= \$target"

Opac window: "\$wind"

Attributes

Attribute	Data Type	Default Value	Possible Values
opac-header-text	text	“In task \$task, for target = \$target, select choice:”	Any text; (\$) references will be resolved
opac-choice-1-description	text	““	Any text; (\$) references will be resolved
opac-choice-2-description	text	““	Any text; (\$) references will be resolved
opac-choice-3-description	text	““	Any text; (\$) references will be resolved
opac-choice-4-description	text	““	Any text; (\$) references will be resolved
opac-timeout	number	-1	Any number. If a default decision should be taken after a certain time period, specify it here, using appropriate time specifications:
opac-default-decision	number	4	The decision (1, 2, 3, or 4) to be taken in case of a timeout or error condition.

2-Way Pattern Decision



The 2-Way-Pattern-Decision block provides two-way branching capability. Branching is based on the comparison of the items on the top of the token stack against the patterns specified in the Choice 1 Pattern and Choice 2 attributes, respectively.

When the token is input to the block, Integrity attempts to match the specified source (for example, the input token stack) against the specified pattern in choice 1. If a match is made, the token is output through the choice 1 output. If no match is made, the source is then compared against the pattern specified in choice 2. If a match is made with this specified pattern, the token is output through the choice 2 output. Thus, the block acts similarly to a case statement

Specify the source to compare in the Source text attribute. Substitution variables are allowed.

The Default Decision attribute allows you to specify the default output that the token will route through if no pattern match is made.

Specify the first pattern to check in the Choice 1 Pattern attribute. Specify the second pattern to check in the Choice 2 Pattern attribute.

OPAC recognizes pattern match language. It is modeled for simplicity after the methods of specifying wild cards in UNIX and DOS as follows:

- An asterisk (*) means 0, 1, 2, or more arbitrary characters will match that position in the pattern. For example:

*PA

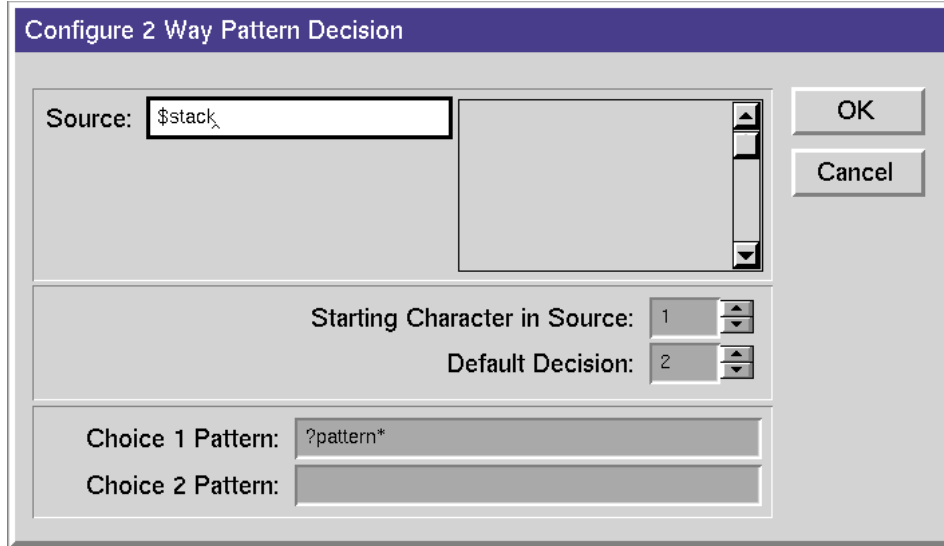
would be matched by OPAC, GRANDPA, and PA.

- A question mark (?) must match exactly one arbitrary character at that position. For example:

? PA?

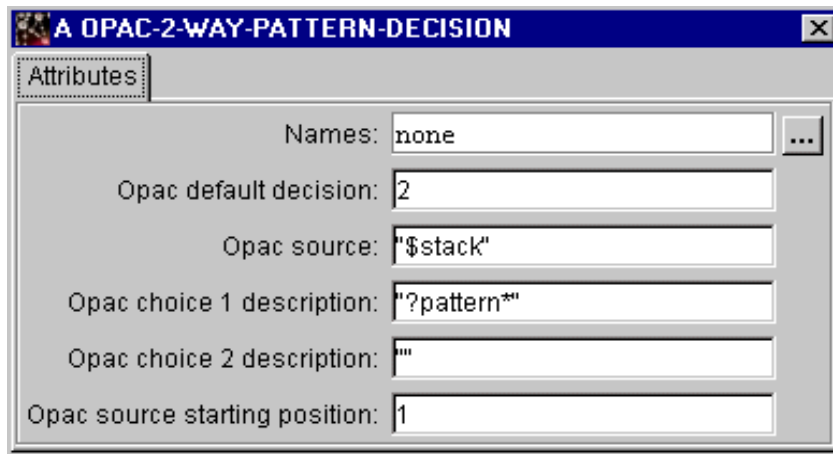
would be matched only by OPAC.

Although substitution variables are allowed in the OPAC Source attribute, they are not allowed in any of the pattern matching strings.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-source	text	"\$stack"	Any text
opac-source-starting-position	number	1	Any number from 1 to the length of the source.

Attribute	Data Type	Default Value	Possible Values
opac-default-decision	number	2	1 or 2
opac-choice-1-description	text	"?pattern*"	Any text, where ? represents one character and * represents any sequence of characters.
opac-choice-2-description	text	""	Any text

3-Way Pattern Decision



The 3-Way Pattern Decision block is similar to the 2-Way Pattern Decision block, except that it offers three pattern choices. The additional choice is specified in the Choice 3 Pattern attribute. The statement acts similarly to a case statement, first checking Choice 1 Pattern, then Choice 2 Pattern, and finally the Choice 3 Pattern. If none of these match, or if an error condition occurs, the default decision is taken.

This block may also be used to represent a two-way decision, with the third choice being the default decision that would be taken only in case of an error condition such as a timeout.

The third choice can also be an input to another pattern decision block, thus extending the number of case statements.

Configure 3 Way Pattern Decision

Source:

Starting Character in Source:

Default Decision:

Choice 1 Pattern:

Choice 2 Pattern:

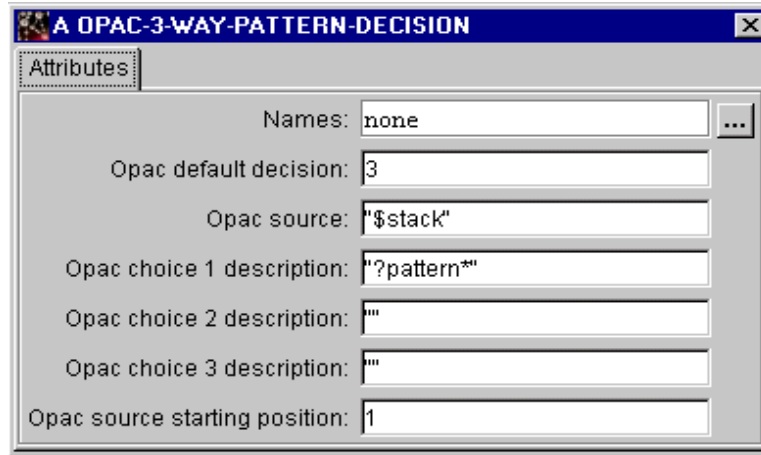
Choice 3 Pattern:

OK

Cancel

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-source	text	“\$stack”	Any text
opac-source-starting-position	number	1	Any number from 1 to the length of the source.
opac-default-decision	number	3	1, 2, or 3
opac-choice-1-description	text	“?pattern*”	Any text, where ? represents one character and * represents any sequence of characters.
opac-choice-2-description	text	“”	Any text
opac-choice-3-description	text	“”	Any text

4-Way Pattern Decision



The 4-Way Pattern Decision block is similar to the 2-Way Pattern Decision block, except that it offers four pattern choices. The additional choices are specified in the Choice 3 Pattern and Choice 4 Pattern attribute.

The statement acts like a case statement, first checking the Choice 1 Pattern, Choice 2 Pattern, Choice 3 Pattern, then the Choice 4 Pattern. If none of these match, or if an error condition occurs, the default decision is taken.

This block can also be used to represent a three-way decision, with the fourth choice being the default decision; this default should be taken only in case of an error condition such as a timeout.

The fourth choice can also be an input to another pattern decision block, thus extending the number of case statements.

Configure 4 Way Pattern Decision

Source:

Starting Character in Source:

Default Decision:

Choice 1 Pattern:

Choice 2 Pattern:

Choice 3 Pattern:

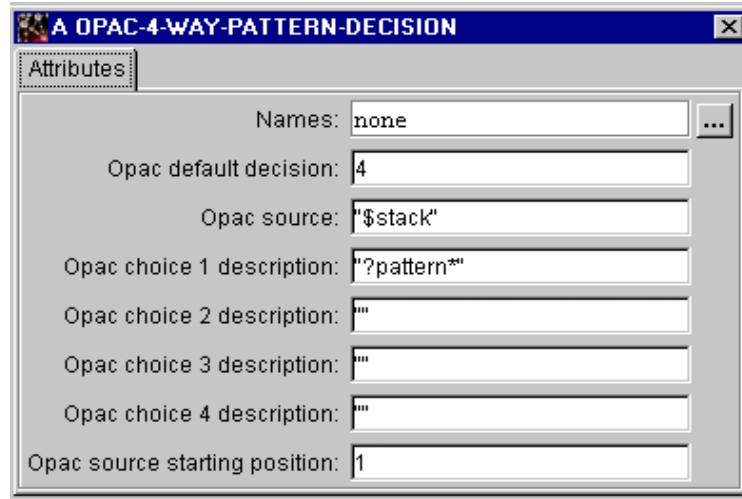
Choice 4 Pattern:

OK

Cancel

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-source	text	“\$stack”	Any text
opac-source-starting-position	number	1	Any number from 1 to the length of the source.
opac-default-decision	number	4	1, 2, 3, or 4
opac-choice-1-description	text	“?pattern*”	Any text, where ? represents one character and * represents any sequence of characters.
opac-choice-2-description	text	“”	Any text

Attribute	Data Type	Default Value	Possible Values
opac-choice-3- description	text	""	Any text
opac-choice-4- description	text	""	Any text

2-Way Pattern Decision By Symbol



The 2-Way Pattern Decision By Symbol block provides two-way branching capability based on specified parsing of an input string into one or more symbols.

The block parses multiple symbols in the input text string sequentially, one symbol at a time. This behavior differs from the behavior of other 2-way decision blocks, which search the entire text for a specified string. The 2-Way Pattern Decision By Symbol block tokenizes the text into parsable symbols, and matches as many as you specify. Symbols are separated by spaces in the Choice Pattern attributes.

Wild cards are also allowed. For example, you could also parse for `"* * * *"` and send the fourth word to some local variable for later use.

Substitution variables are not allowed in the pattern match string.

If a match is made with the Choice 1 Pattern, the token is output through port 1, and that parsable symbol is removed from the text on the stack. If no match occurs, the token is output through port 2, and the stack is left unchanged.

The block uses text that is on the stack; it does not use any other source.

A common use for this block is to write parsed symbols to local variables for later use in a procedure.

An arrangement of these blocks can be used to accomplish arbitrary n-symbol look-ahead parsing. Arranging a series of these blocks vertically, each block would represent a successful parse. If a match fails, the stack is still intact and another match is tried, regardless of the number of symbols you tried to match that are separated by spaces in the pattern specification in the Choice 1 Pattern.

If the Send Parsed Result To attribute is set, OPAC sends the most recently discovered symbol to the specified location.

As an example, you can check for the first four words of "This is a sample document for parsing", by setting the Choice 1 Pattern of the block to:

"This is a sample."

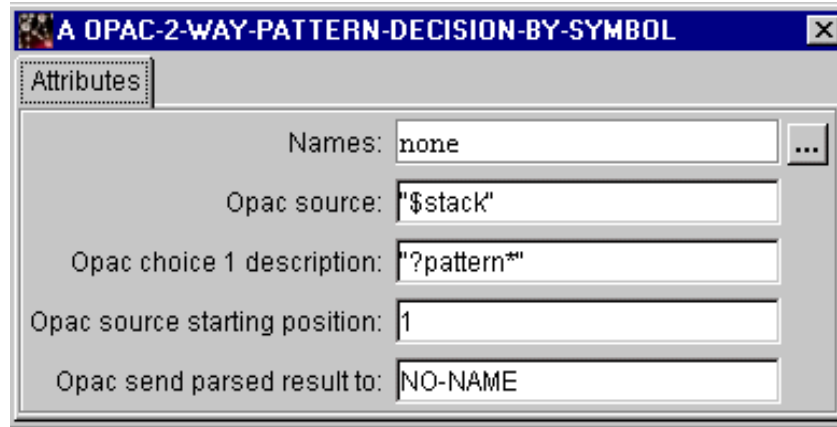
The image shows a dialog box titled "Configure 2 Way Pattern Decision". It has a source text field containing "\$stack", a starting character spinner set to "1", a default decision spinner set to "2", a choice 1 pattern text field containing "?pattern*", and an empty choice 2 pattern text field. "OK" and "Cancel" buttons are on the right.

Attributes

Attribute	Data Type	Default Value	Possible Values
opac-source	text	"\$stack"	Must be \$stack
opac-send-parsed-result-to	symbol	no-name	The location where the most recently discovered symbol should be sent.
opac-source-starting-position	number	1	Any number from 1 to the length of the source.
opac-choice-1-description	text	"?pattern*"	Any text, where ? represents one character and * represents any sequence of characters.

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-decision-proc	symbol	opac-consume-decision-from-stack	User-defined G2 procedure with arguments.
opac-choice-1-description	text	""	Any text; (\$) references will be resolved
opac-choice-2-description	text	""	Any text; (\$) references will be resolved
opac-choice-3-description	text	""	Any text; (\$) references will be resolved
opac-choice-4-description	text	""	Any text; (\$) references will be resolved

Attribute	Data Type	Default Value	Possible Values
opac-timeout	number	-1	Any number. If a default decision should be taken after a certain time period, specify it here, using appropriate time specifications.
opac-default-decision	number	4	The decision (1, 2, 3, or 4) to be taken in case of a timeout or error condition.

Operating System (OS) Palette

Describes the blocks in the OS Actions palette.

Introduction	85
Set Local Integer From Source	87
File Exists Test	88
Delete File	90
Kill Process	92
Spawn Return Output	94
Spawn Return PID	96
Spawn No Return	98
Write File	99
Read File	101



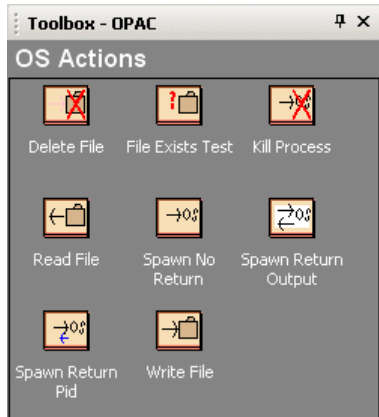
Introduction

The Integrity graphical language provides blocks representing standard interactions with the underlying operating system. OPAC currently supports UNIX and Windows when making calls outside of the machine running Integrity

Standard operations are supported such as reading, writing, and finding files, and spawning processes and getting the returned information. You can also replace UNIX scripts or Windows batch files with these blocks. UNIX-style substitution variables are supported, such as prefacing an argument with (\$).

Note When referencing any filename or directory, it is up to the user to provide the correct directory delimiter, such as (\) for Windows or (/) for UNIX.

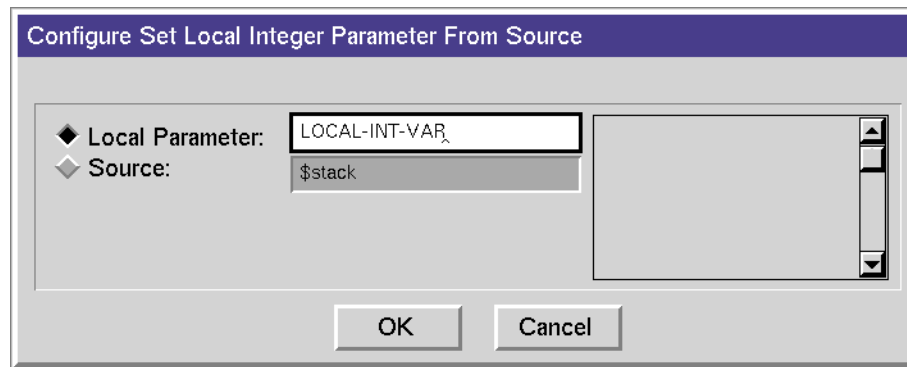
Here is the OS Actions palette:



Set Local Integer From Source

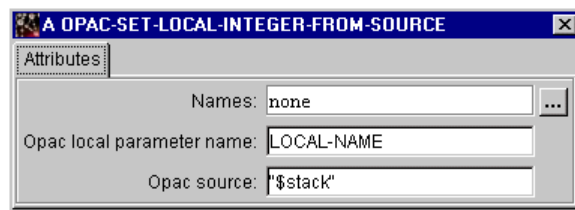


Use the Set Local Integer From Source block to set the specified local integer parameter specified to the value specified by the source. (for example, the stack). Substitution variables are allowed.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-local-parameter-name	symbol	local-int-var	Any opac-local-integer-parameter
opac-source	text	"\$stack"	stack, \$stack. Other (\$) references will be resolved.

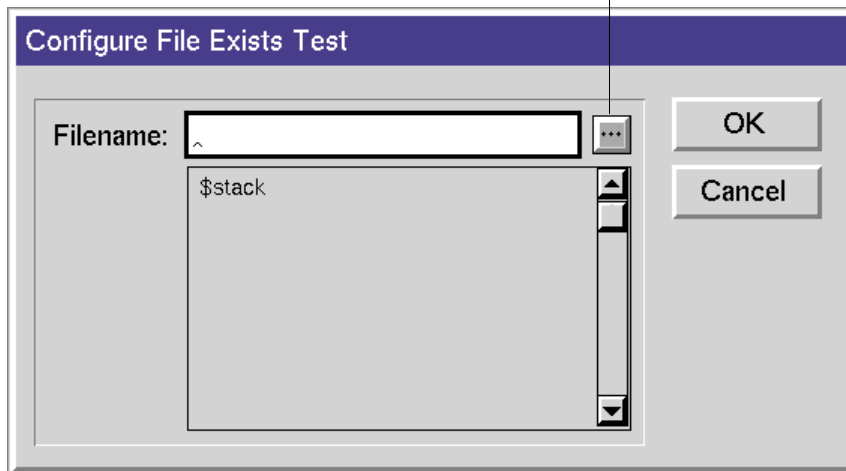
File Exists Test



Use the File Exists Test block to see if a file exists. Specify the file, including the file path and file name, to test for in the OPAC Filename attribute.

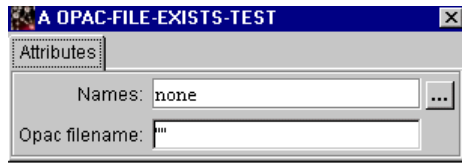
To browse a list of files from which to choose a file for the test, click on the Browse Directory button indicated by the ellipsis.

Browse Directory Button



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-filename	text	""	Any valid file path and filename. Make sure the appropriate directory/file separator is used for current OS. (\$) references will be resolved.

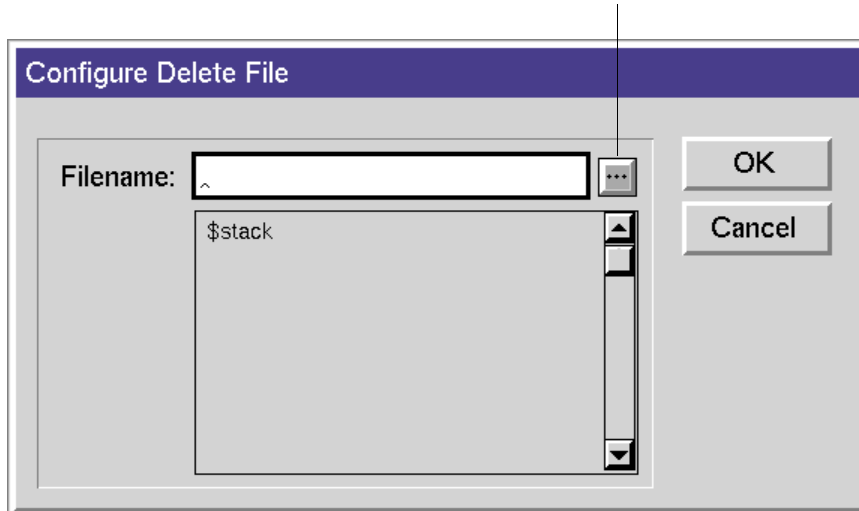
Delete File



Use the Delete File block to delete a file from the file system. Specify the file to delete, including the file path and file name, in the OPAC Filename attribute.

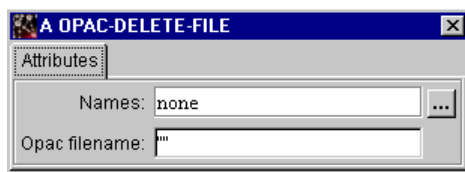
To browse a list of files from which to choose a file for deletion, click on the Browse Directory button indicated by the ellipsis.

Browse Directory Button



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

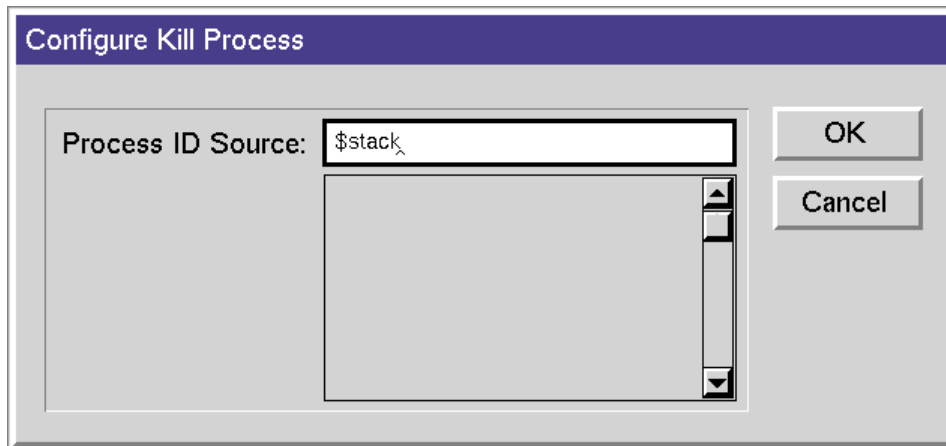
Attribute	Data Type	Default Value	Possible Values
opac-filename	text	""	Any valid file path and filename. Make sure the appropriate directory/file separator is used for current OS. (\$) references will be resolved.

Kill Process



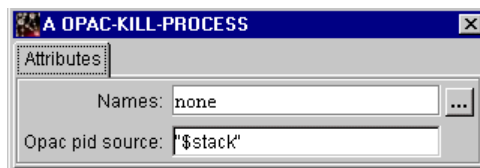
Use the Kill Process block to kill a process in the UNIX operating system. The source of the process ID for the process is specified in the OPAC PID Source attribute. This source is either the Stack of the token, or can be a local variable. See [Spawn Return PID](#).

Caution If you use the PID returned from the Spawn Return PID block on a Windows platform as input to the Kill Process block, you will not kill the process. The PID returned on a Windows platform is the PID of the command shell. Using the block generates an error because the command shell that initiates the process terminates.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-pid-source	symbol	stack	Stack or an opac-local-integer-parameter.

Spawn Return Output



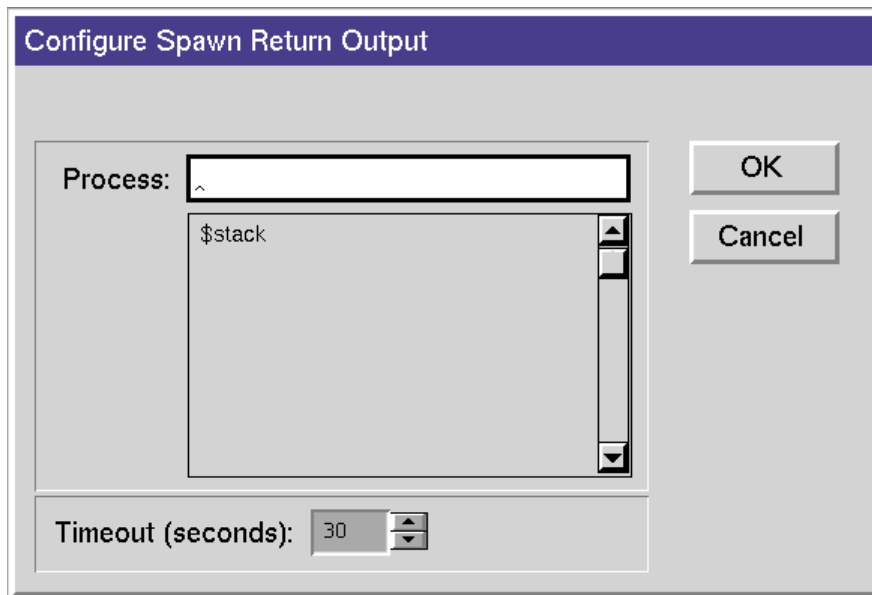
The Spawn Return Output block is used to spawn a process to the operating system and return the resulting output to the specified source. For example, in the UNIX operating system, you can spawn a (`ls -l`) command to get a list of the files and return them to OPAC for further analysis. As in other spawn commands, the block is similar to specifying a command at a command line prompt.

Caution Do not embed a carriage return. An embedded carriage return causes the Spawn Return Output block to terminate.

Specify the command line entry in the Process attribute of the the Configure dialog. (This attribute is the OPAC Spawn Spec attribute in the Propertied Dialog box.) The results are returned to the stack. Each line returned is placed as a separate entry in a text list. (In general, text files are converted to text lists in OPAC.)

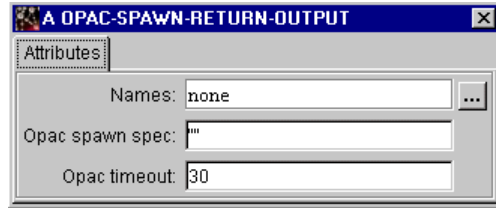
The OPAC Timeout attribute specifies the number of seconds to wait for results to return. If the specified timeout is exceeded, the token is flagged with an error, and the token continues.

If a failure to get return information is significant, you can use an If Token Error Free block (an instance of the 2-Way Decision block) to check the token, either after the Spawn block, or perhaps later in the code before a significant action occurs.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-spawn-spec	text	""	Any valid command for the current OS. (\$) references will be resolved.
opac-timeout	integer	30	Maximum number of seconds to wait for the returned output

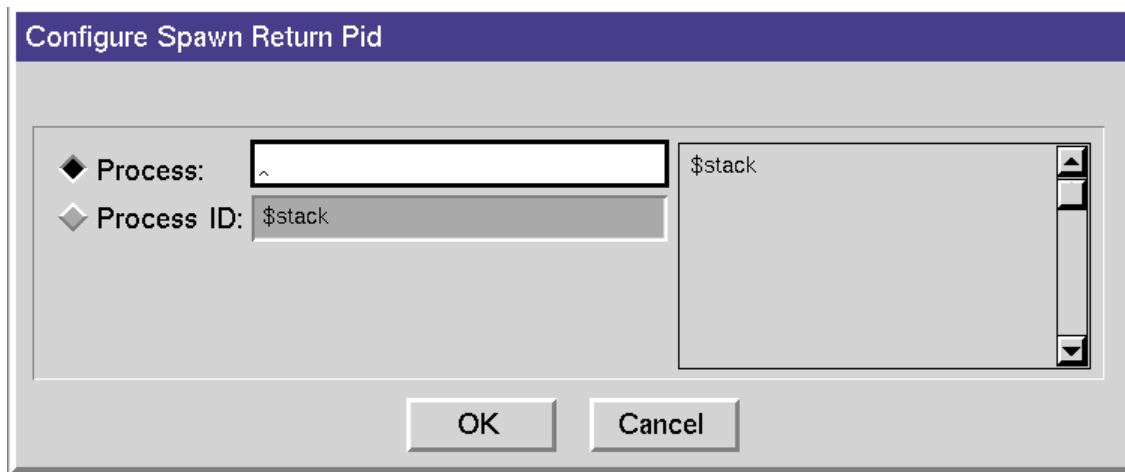
Spawn Return PID



The Spawn Return PID block is used to spawn a process to the operating system. In the UNIX operating system, the block also returns the resulting process ID (PID). You may need this PID to later kill the process. As in other spawn commands, this block is similar to specifying a command at a command line prompt.

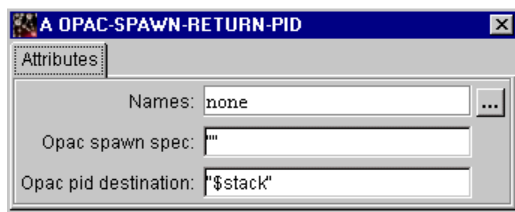
The command line entry is specified in the Process attribute of the the Configure dialog. (This attribute is the OPAC Spawn Spec attribute in the Propertied Dialog box.) The results are returned to the destination specified in the Process ID attribute in the Configure Dialog box. (This attribute is the OPAC PID Destination attribute in the Properties Dialog Box.) This is normally the Stack or a local variable.

Note Using this block in the Windows operating system does not return the PID of the process you specify. Instead, it returns the PID of the command shell in which the specified process was spawned. Do not use this PID in the Kill Process block, as doing so terminates the entire command shell.



Properties Dialog

The Properties dialog for the block is shown below:



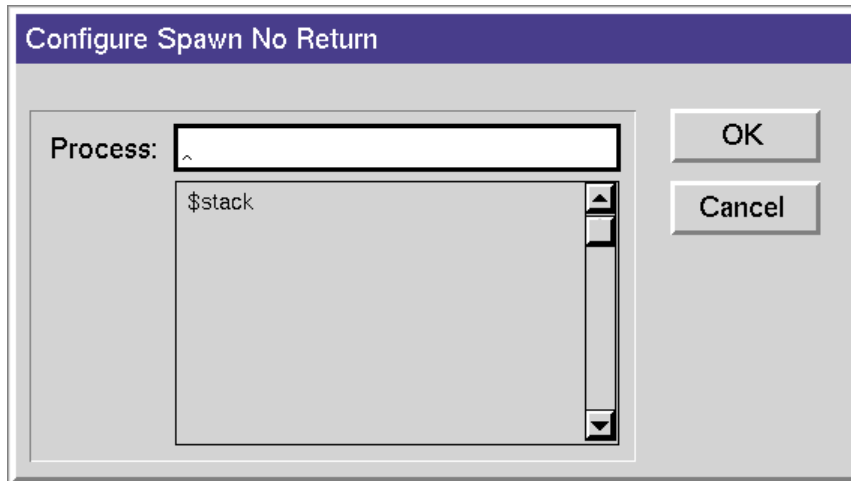
Attributes

Attribute	Data Type	Default Value	Possible Values
opac-spawn-spec	text	""	Any valid command for the current OS. (\$) references will be resolved.
opac-pid-destination	symbol	stack	Stack or an opac-local-integer-parameter.

Spawn No Return

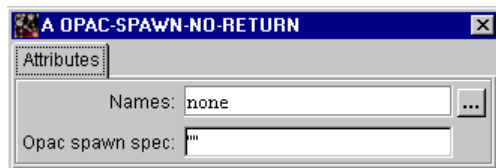


The Spawn No Return block spawns the specified process to the operating system. No information is returned about the spawned process. This block is equivalent to specifying a command at a command line prompt. Specify the command line entry in the Process attribute of the Configure dialog or in the OPAC Spawn Spec attribute of the Properties dialog.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

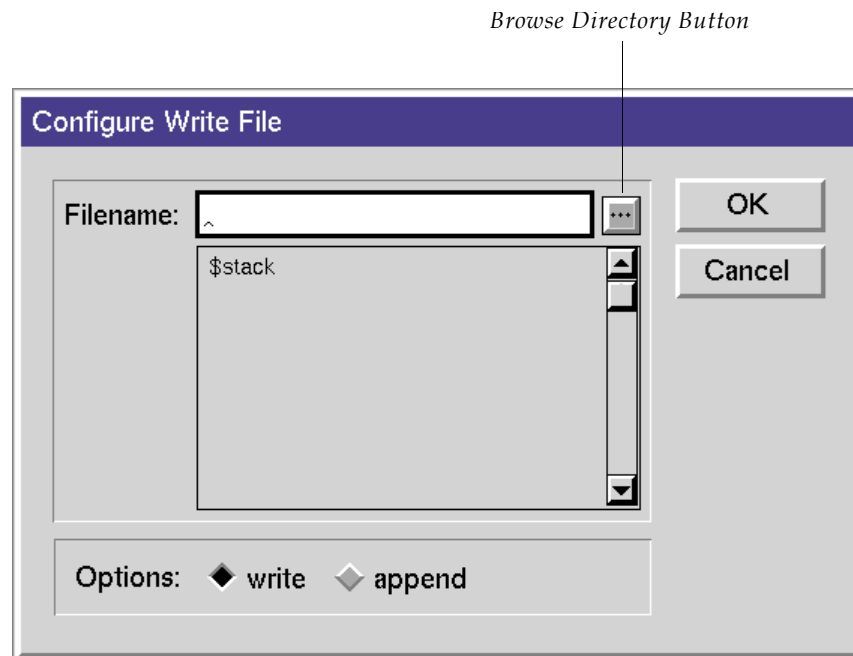
Attribute	Data Type	Default Value	Possible Values
opac-spawn-spec	text	""	Any valid command for the current OS. \$references will be resolved.

Write File



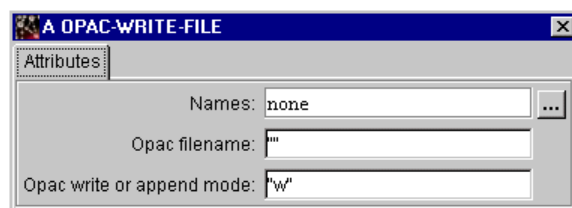
The Write File block writes the text at the top of the token stack to a file specified by the OPAC Filename attribute. The source of the file content is taken from the stack and is either a text-list or text. If the source is a text-list, each text in the text-list corresponds to one line in the output file.

The attribute OPAC Write or Append Mode specifies whether to overwrite (w) or append (a) to the specified file, if it exists. If the specified file does not exist, it is created.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-filename	text	""	any valid file path and filename. Make sure the appropriate directory/file separator is used for current OS. (\$) references will be resolved.
opac-write-or-append-mode	text	"w"	"w" - write to file "a" - append to file

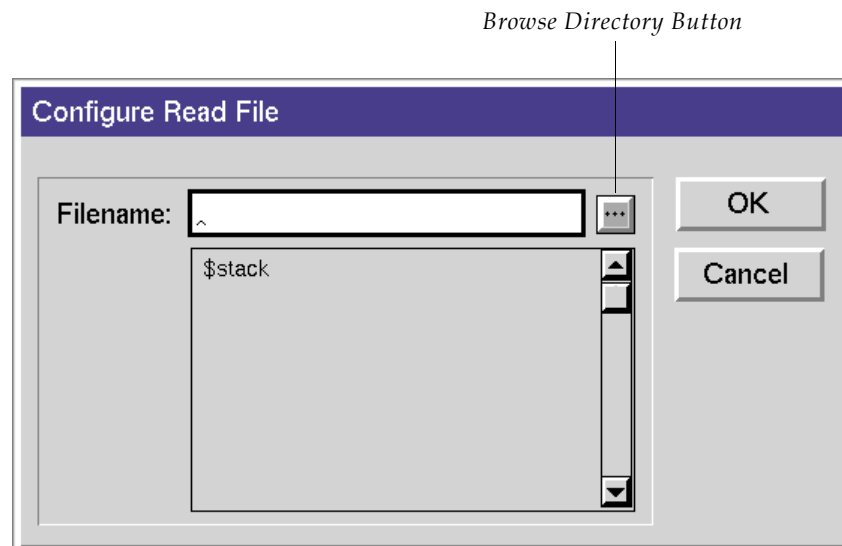
Read File



The Read File block reads a file, specified by the OPAC Filename attribute and places the result on the stack as a text list. Each text in the text list corresponds to one line in the input file.

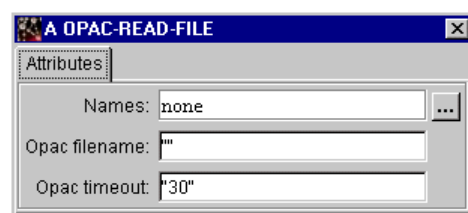
Make sure that the OPAC Filename attribute does not point to an empty file. If the attribute does point to an empty file, the procedure will suspend execution of the token and timeout after 30 seconds.

Note If you are using Windows, this procedure may have difficulty reading a line with a backslash (\). This OPAC procedure uses the underlying G2 System procedure G2-read-line. In this version of G2 on the Windows platform, the backslash is the escape code; any character after the backslash is interpreted as the escape sequence.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-filename	text	""	Any valid file path and filename. Make sure the appropriate directory/file separator is used for current OS. (\$) references will be resolved.

Stack Operations Palette

Describes the blocks in the Stack Operations palette.

Introduction	103
Generic Put Something On Stack	105
Pop General Stack	107
Put Connected Objects On Stack	108
Pop General Stack And Delete	110
Put Item On Stack	111
Put Float On Stack	112
Put Integer On Stack	113
Put Text On Stack	114

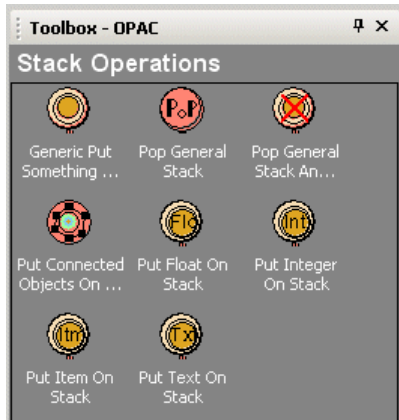


Introduction

The OPAC Tokens maintain a stack for general use. It is possible to place items on this stack, operate on those items in subsequent blocks, and then delete those items from the stack in another block.

Note Certain blocks consume the stack (remove an item from the stack). The description indicates whether a block consumes the stack.

Here is the Stack Operations palette:



Using local parameters instead of stack operations, is usually recommended:

- References to named local parameters are clearer than references to a stack.
- Using local parameters allows easier access if several local parameters are required, rather than just one item at the top of the stack.
- Some blocks may consume elements of the stack, while others do not, so that the exact behavior is not quite so obvious from the diagram.

However, in many cases the easier design is to get an item from one block, put it on the stack, and immediately use the stack in the next block. This practice reduces the number of blocks. Passing items from one block to the block immediately following is the preferred application of stack operations.

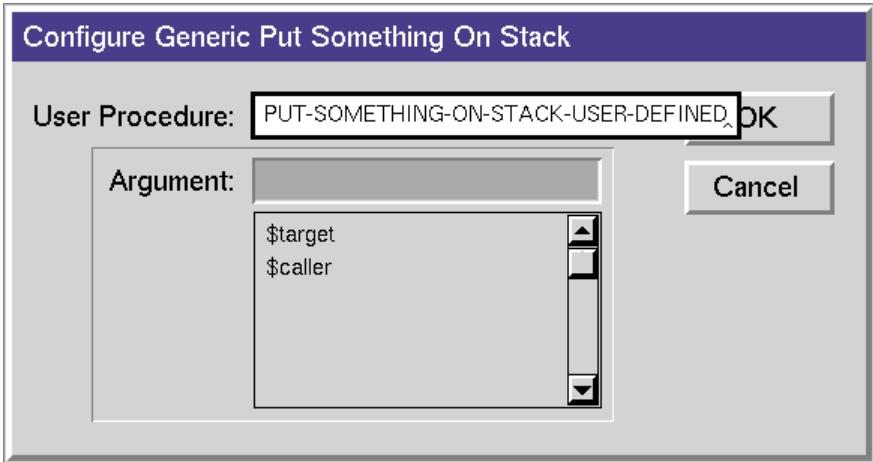
Note The stacks for each token are completely independent of each other. No interactions exist between the tokens, even if they are executing the same graphical procedure.

Generic Put Something On Stack



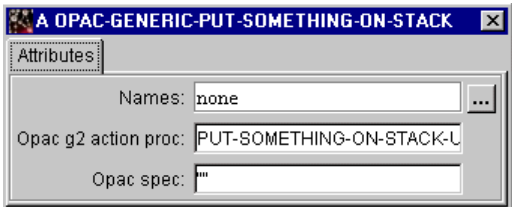
The Generic Put Something On Stack block calls a user-defined procedure to place something on the stack. Specify the user-defined G2 procedure in the User Procedure attribute of the Configure dialog box or in the OPAC G2 action proc attribute of the Properties dialog box.

If the specified procedure does not exist, you can use this block to create a new Procedure block. To create the new Procedure block, click on the ellipsis button. The Configure Create New Procedure dialog box appears. Enter the name of the new procedure in the Procedure Name attribute, then click on the OK button. This action creates a Procedure block and displays the new block on the same workspace with the Generic Put Something On Stack block. This new Procedure block includes a skeleton procedure that you may edit. To set the properties of this Procedure block and edit the text of the procedure, see the section "Procedure Block" in the "General Actions" chapter.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attributes	Data Type	Default Value	Possible Values
opac-g2-action-proc	symbol	put-something-on-stack-user-defined	Any G2 procedure.
opac-spec	text	""	Can be used by the procedure specified in the opac-g2-action-proc.

Pop General Stack



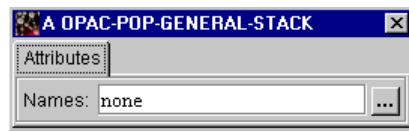
The Pop General Stack block pops the general stack of token. (To “pop the stack” means to remove the top item from the stack.)

A related block, the Pop General Stack And Delete block, pop the stack and deletes the item. The Pop General Stack block does not delete the item, only its reference in the stack.

Note Use this block carefully. A common mistake is to omit deleting items popped from the stack, resulting in an excess of items that are no longer used. This block is used usually for objects that are permanent, or members of other lists.

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attributes	Data Type	Default Value	Possible Values
N/A			

Put Connected Objects On Stack

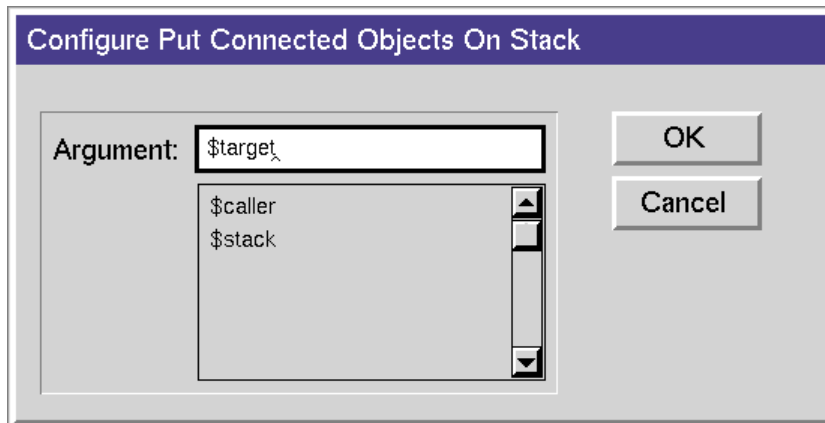


The Put Connected Objects On Stack block generates an item-list, without duplicates, of objects connected to the objects specified by the Argument attribute of the Configure dialog box or in the OPAC Spec attribute of the Properties dialog. The argument can reference a list, in which case this block finds all objects connected to those objects in the specified list.

The procedure places in a list on the stack all objects that are connected to the specified object.

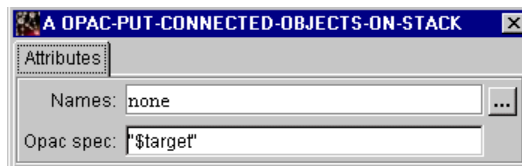
The OPAC Spec default value is `$target`. The procedure also allows `$stack` to be specified in the OPAC Spec attribute. In that case, the procedure finds all objects connected to the specified object in the stack. If the stack contains an item-list of objects, it returns an item-list of all objects connected to all of those objects.

Placing two of these blocks in sequence can find all objects connected within a distance of two objects from the originally specified one. Placing three blocks in sequence can find all objects connected within a distance of three objects, and so on. The list that is generated allows only a single entry for a given object. This technique is illustrated in the OPAC Demo Chapter of the *Integrity User's Guide*.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-spec	text	""	Any object or an opac-local-item.

Pop General Stack And Delete



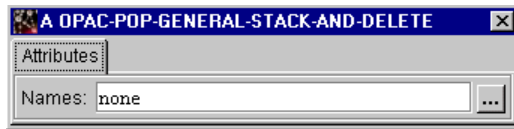
The Pop General Stack And Delete block pops the general stack (removes the top item from the stack) and deletes the referenced item.

A related block, the Pop General Stack block, pops the stack, but does not delete the item. The Pop General Stack block deletes only the item reference in the stack.

Caution The Pop General Stack And Delete block does not delete permanent items.

When a list is the item to be popped and deleted, all list items are also deleted.

The Properties dialog for the block is shown below:



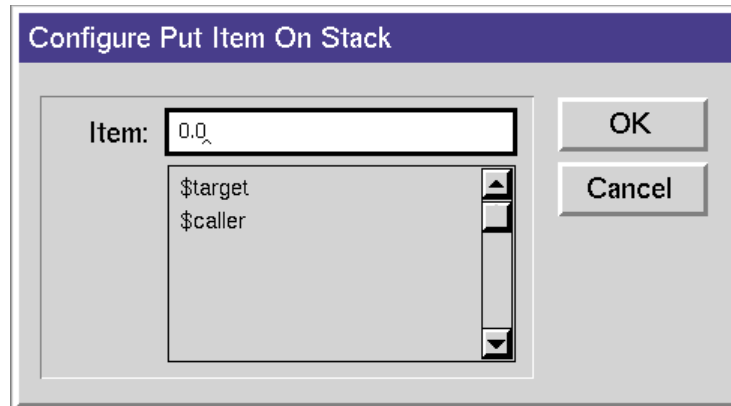
Attributes

Attribute	Data Type	Default Value	Possible Values
N/A			

Put Item On Stack

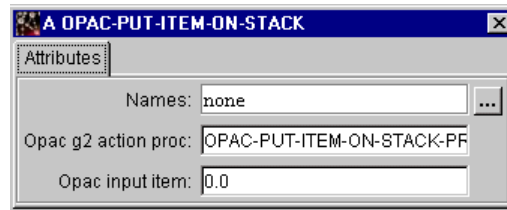


The Put Item On Stack block takes a value from the item specified by the Input Item attribute and places it on the stack for the token. The block allows substitution variables within the specification.



Properties Dialog

The Properties dialog for the block is shown below:



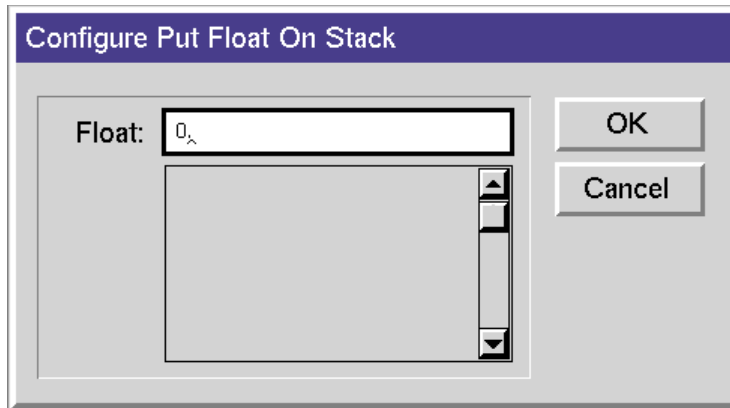
Attributes

Attribute	Data Type	Default Value	Possible Values
opac-input-item	value	0.0	Any value.

Put Float On Stack

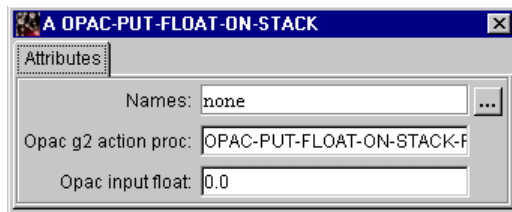


The Put Float On Stack block takes a floating point value from the Input Float attribute and places it on the token stack. The block allows substitution variables within the specification.



Properties Dialog

The Properties dialog for the block is shown below:



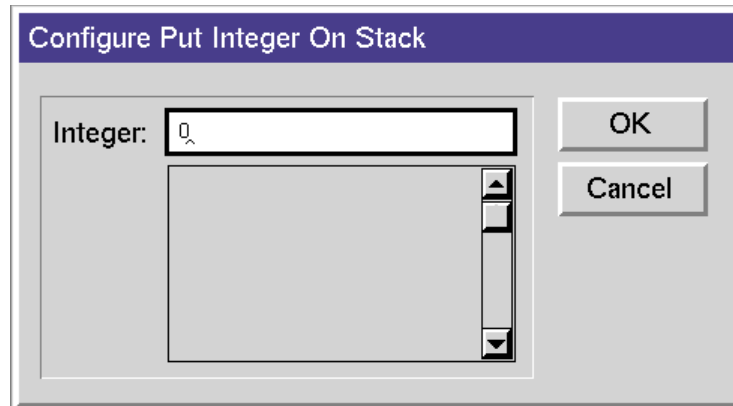
Attributes

Attribute	Data Type	Default Value	Possible Values
opac-input-float	float	0.0	Any float.

Put Integer On Stack

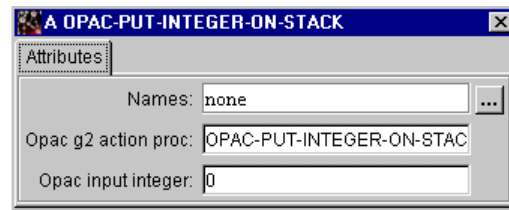


The Put Integer On Stack block takes an integer value from the Input Integer attribute and places it on the token stack. The block allows substitution variables within the specification.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

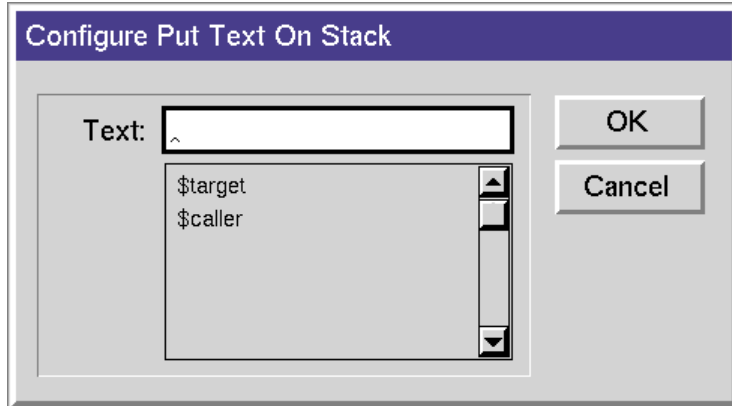
Attribute	Data Type	Default Value	Possible Values
opac-input-integer	integer	0	Any integer.

Put Text On Stack



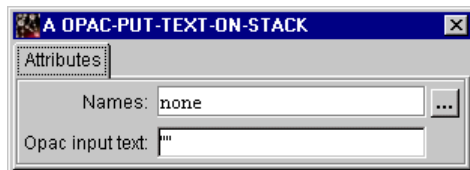
The Put Text On Stack block places text from the Input Text attribute on the token stack. The block allows substitution variables within the specification.

Note OPAC Input Text cannot be in the form: the <attribute> of <instance>.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-input-text	text	""	Any text value. (\$) references will be resolved.

Local Parameters Palette

Describes the blocks in the Local Parameters palette.

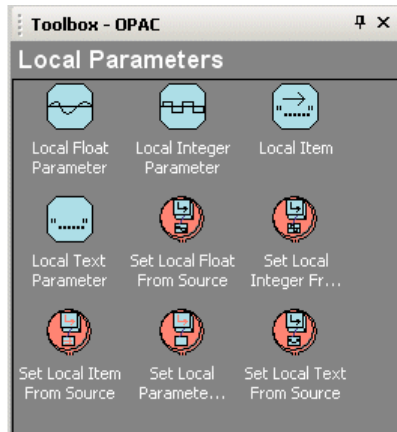
Introduction	116
Set Local Float From Source	117
Local Float Parameter	119
Set Local Item From Source	121
Local Item	122
Set Local Integer From Source	124
Local Integer Parameter	125
Set Local Text From Source	127
Local Text Parameter	128



Introduction

This chapter describes using parameters. Parameters hold data that can be accessed and manipulated by multiple OPAC blocks within the same OPAC procedure. These parameters are accessible from the Subtask Start through the Subtask End.

There is the Local Parameters palette:



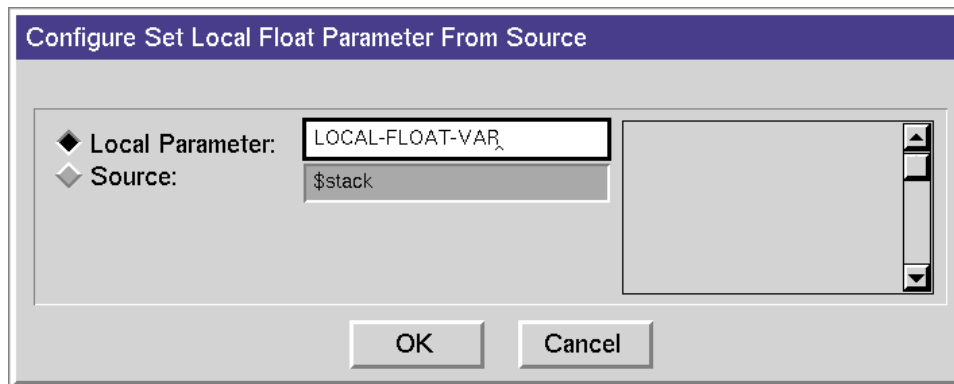
Use the OPAC Parameters to define and set parameters for a given Subtask Start block. You must attach these parameters to the Subtask Start block. Supporting blocks for setting these parameters also exist.

Set Local Float From Source



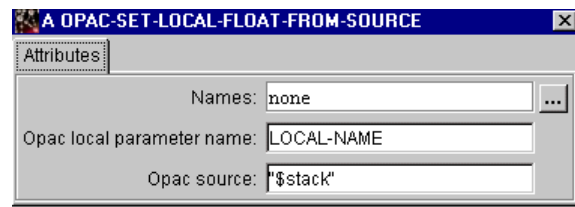
Use the Set Local Float From Source block to set the local float parameter specified by the Local Parameter attribute to the value specified as the source. The source can be:

- \$stack
- a value
- a substitution variable



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-local-parameter-name	symbol	local-name	Any local-float-parameter
opac-source	text	“\$stack“	stack, \$stack. Other (\$) references will be resolved.

Local Float Parameter



Use the Local Float Parameter block to define local float parameters for a given Subtask Start block, the entry point for the subtask. These blocks must be attached directly to the Subtask Start block, the entry point for the subtask.

The Initial Value attribute of these parameters determines the initial value for the variable of a given token entering that procedure. When the token enters the procedure, it clones one of these blocks, and keeps its own local copy. Thus, normal G2 parameter initialization is used. Each token, as a result of encountering various OPAC blocks, may change the value of its local copy of local parameters.

Tokens do not interact. The next token to enter this subtask starts with the initial value when it clones its copy of this block.

If arguments are passed to a subtask, and if the local parameters attached to the Subtask Start block match those specified in the OPAC arguments of the Subtask Start block, OPAC will insert the values in the new copies of this block, specific to each Token (see [Configuring Arguments Blocks](#) and [Creating OPAC Procedure Arguments From a G2 Procedure](#)).

Properties Dialog

The Properties dialog for the block is shown below:

Attributes

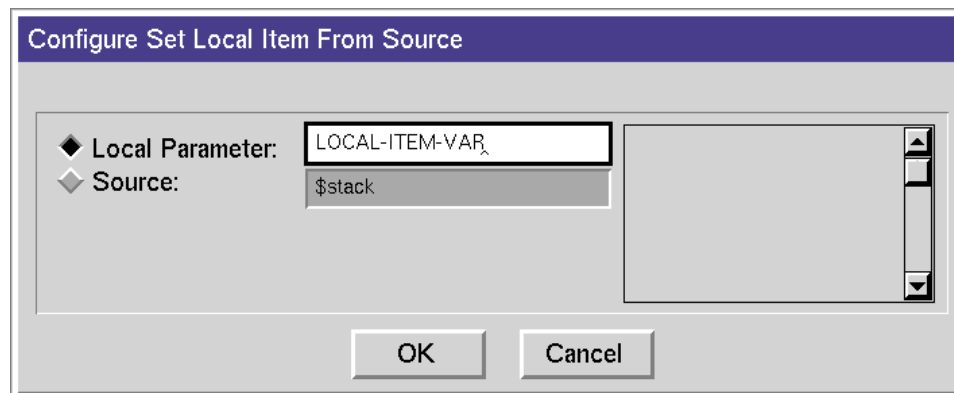
Attribute	Data Type	Default Value	Possible Values
opac-local-name	symbol	local-name	Any local parameter name.
initial-value	float	0.0	Any float value.

Set Local Item From Source



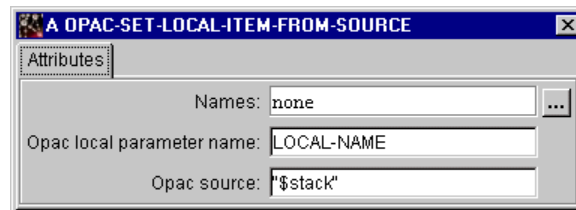
The Set Local Item From Source block sets the specified local parameter to the value specified by the source. The Set Local Item From Source block removes the current relation, if one exists, from and assigns a new relation to the item specified as the source.

When the Source attribute of the Set Local Item From Source block is `$stack` and the stack is empty, the OPAC Local Item is set to the token.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-local-parameter-name	symbol	local-item-var	Any opac-local-item
opac-source	text	"\$stack"	stack, \$stack. Other (\$) references will be resolved.

Local Item



Use the Local Item block to define local items for a given subtask. These blocks must be attached directly to the Subtask Start block, the entry point for the subtask.

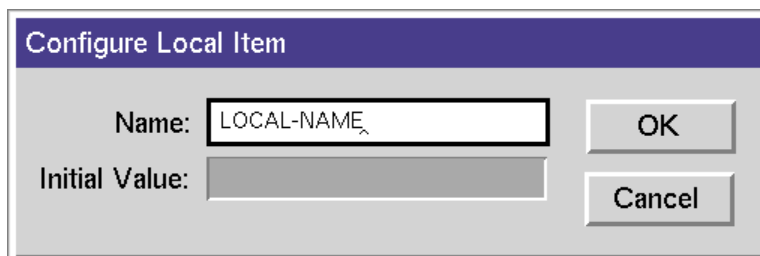
The blocks represent pointers to objects (indicated by the arrow on the icon). You can refer to the items either directly by name or indirectly, using substitution variables.

The Initial Value attribute of these parameters determines the initial value of the name reference for each token entering that procedure.

When the token enters the procedure containing one of these blocks, it clones the block, and keeps its own local copy. Thus, normal G2 parameter initialization is used. Each token, as a result of encountering various OPAC blocks, may change the value of its local copy of local parameters.

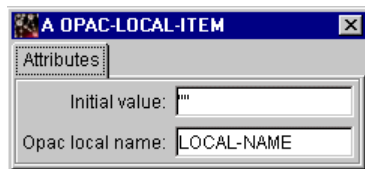
Tokens that subsequently enter the procedure do not interact with previous tokens. The next token to enter this subtask starts with the initial value when it clones its copy of this block.

If arguments are passed to a subtask, and if the local parameters attached to the Subtask Start block match those specified in the OPAC arguments of the Subtask Start block, OPAC will insert the values in the new copies of this block, specific to each Token (see [Configuring Arguments Blocks](#) and [Creating OPAC Procedure Arguments From a G2 Procedure](#)).



Properties Dialog

The Properties dialog for the block is shown below:



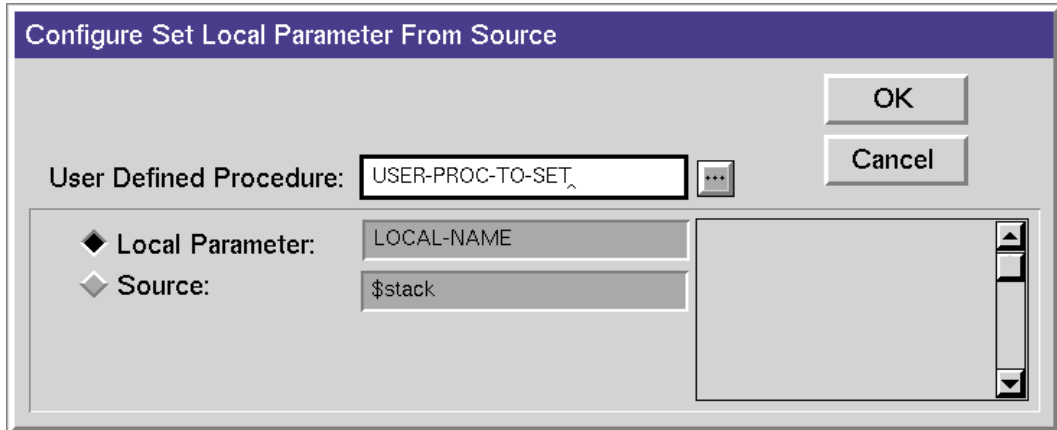
Attributes

Attribute	Data Type	Default Value	Possible Values
opac-local-name	symbol	local-name	Any local item name.
initial-value	text	""	Any text specifying the name of an item.

Set Local Integer From Source

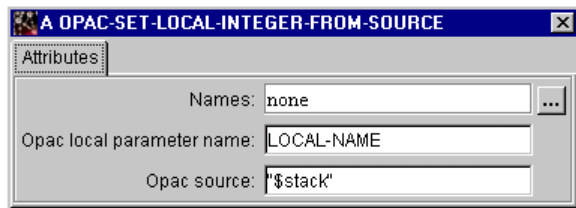


Use the Set Local Integer From Source block to set the local integer parameter specified by the Local Parameter attribute to the value specified by the Source attribute.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-local-parameter-name	symbol	local-int-var	Any opac-local-integer-parameter
opac-source	text	“\$stack”	stack, \$stack. Other (\$) references will be resolved.

Local Integer Parameter



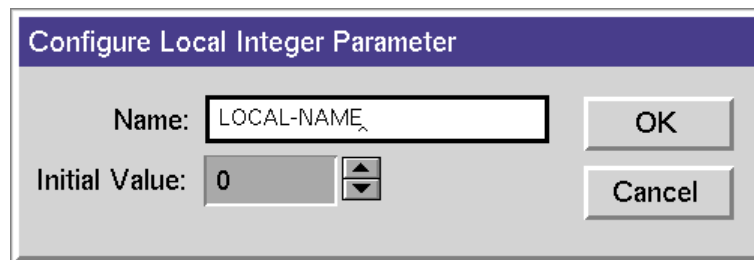
Use the Local Integer Parameter block to define local integer parameters for a given subtask. These blocks must be attached directly to the Subtask Start block, the entry point for the subtask.

The Initial Value attribute of these parameters determines the initial value for the variable of a given token entering that procedure.

When the token enters the procedure containing one of these blocks, it clones the block, and keeps its own local copy. Thus, normal G2 parameter initialization is used. Each token, as a result of encountering various OPAC blocks, may change the value of its local copy of local parameters.

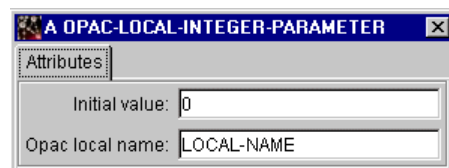
Tokens that subsequently enter the procedure do not interact with previous tokens. The next token to enter this subtask starts with the initial value when it clones its copy of this block.

If arguments are passed to a subtask, and if the local parameters attached to the Subtask Start block match those specified in the OPAC arguments of the Subtask Start block, OPAC will insert the values in the new copies of this block, specific to each Token (see [Configuring Arguments Blocks](#), and [Creating OPAC Procedure Arguments From a G2 Procedure](#)).



Properties Dialog

The Properties dialog for the block is shown below:



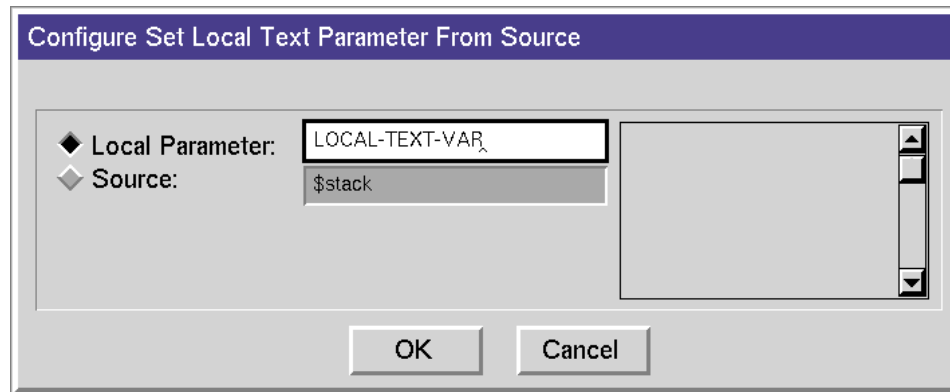
Attributes

Attribute	Data Type	Default Value	Possible Values
opac-local-name	symbol	local-name	Any local parameter name.
initial-value	integer	0	Any integer.

Set Local Text From Source

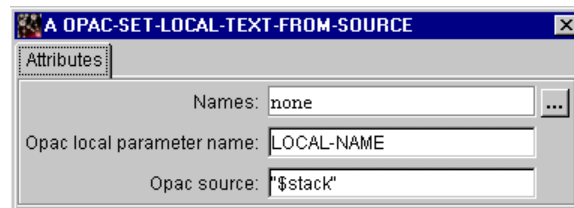


Use the Set Local Text From Source block to set the item specified in the Local Parameter attribute to the value specified in the Source attribute.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-local-parameter-name	symbol	local-text-var	Any opac-local-text-parameter
opac-source	text	"\$stack"	stack, \$stack. Other (\$) references will be resolved.

Local Text Parameter



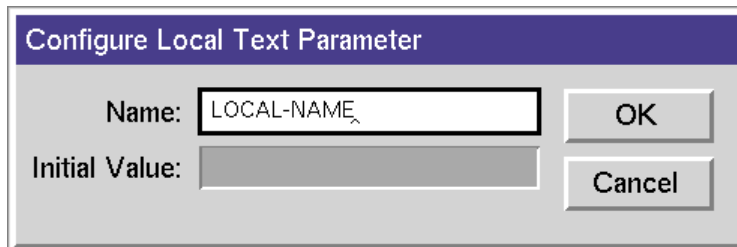
Use the Local Text Parameter block to define local text parameters for a given subtask. Attach these blocks directly to the Subtask Start block, the entry point for the subtask.

The Initial Value attribute of these parameters determines the initial value for the variable of a given token entering that procedure.

When the token enters the procedure containing one of these blocks, it clones the block, and keeps its own local copy. Thus, normal G2 parameter initialization is used. Each token, as a result of encountering various OPAC blocks, may change the value of its local copy of local parameters.

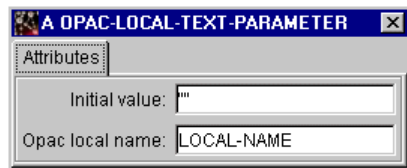
Tokens that subsequently enter the procedure do not interact with previous tokens. The next token to enter this subtask starts with the initial value when it clones its copy of this block.

If arguments are passed to a subtask, and if the local parameters attached to the Subtask Start block match those specified in the OPAC arguments of the Subtask Start block, OPAC will insert the values in the new copies of this block, specific to each token.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-local-name	symbol	local-name	Any local parameter name.
opac-input-text	text	""	Any text.

Subtask Arguments Palette

Describes the blocks in the Subtask Arguments palette.

Introduction 131

Value Argument 132

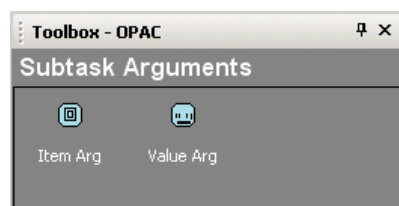
Item Argument 134



Introduction

This chapter describes Subtask Arguments palette blocks. Use Subtask Arguments when you want to pass arguments from one OPAC procedure to another OPAC procedure. You can pass arguments either by passing a specified value or by reference. These arguments are attached to the Macro and Subtask blocks.

Here is the Subtask Arguments palette:



Value Argument



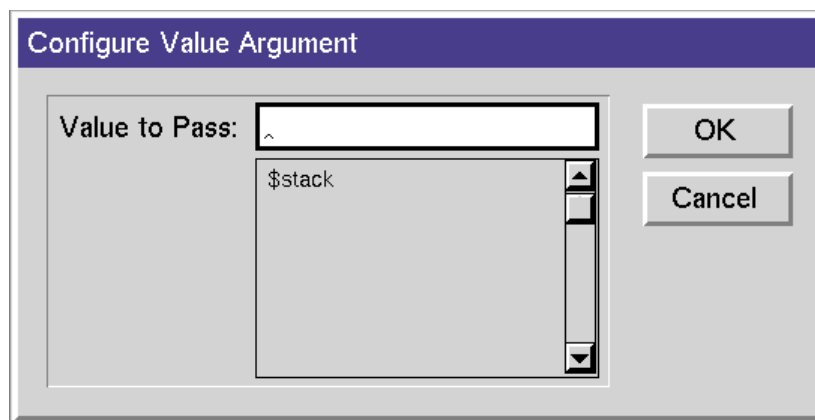
The Value Argument block passes the specified value as an argument to a Subtask.

Use the Value Argument block and the Item Argument blocks to specify passing arguments from one OPAC procedure to another. Argument blocks are chained, with the first argument block attached to the Subtask block that calls the argument. The second argument block is attached to the first argument block, and so on.

These blocks are normally arranged off to the right of the Subtask block. The order is important because as the arguments are placed in the stack, they are removed from the stack when the matching arguments attribute is met from the subtask.

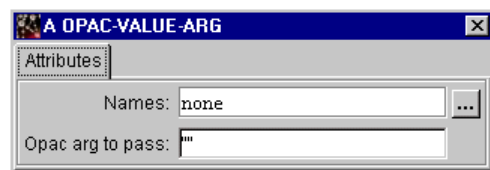
Use the Value Argument block to specify the value of an argument that will be passed. Use the Item Argument block to specify passing an argument by reference. This is explained in a separate topic (see [Passing Arguments to a Subtask](#)).

In the case of a Value Argument block, the value to be passed is specified in the argument Value to Pass attribute of the block



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-arg-to-pass	text		““ opac-local-integer-parameter opac-local-item opac-local-float-parameter

Item Argument



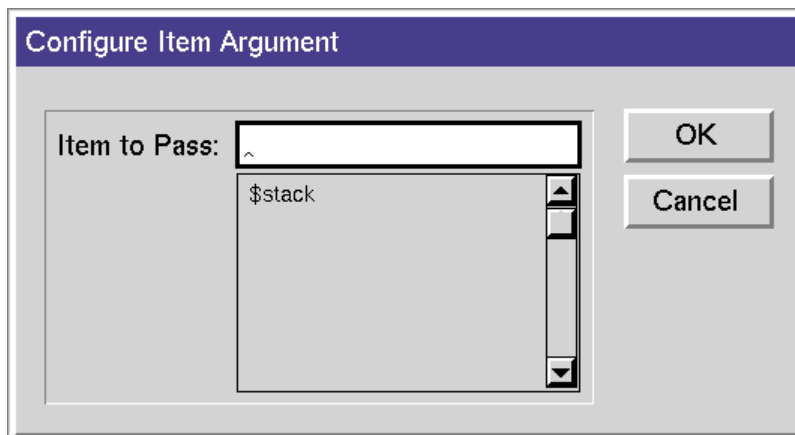
Use the Item Argument block to pass a value specified by the item name as an argument to a Subtask block.

Use the Item Argument block and the Value Argument blocks to specify the passing of arguments to a subtask. The first argument block is attached to the Subtask block to call a subtask. The second argument block is attached to the first argument block, and so on.

These blocks are normally arranged off to the right of the Subtask block. The order is important because as the arguments are placed in the stack, they are removed from the stack when the matching arguments attribute is met from the subtask.

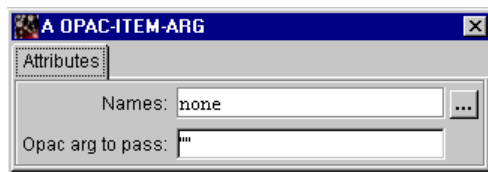
Use the Value Argument block to specify the value of an argument that will be passed. Use the Item Argument block to specify passing an argument by reference. This is explained in a separate topic (see [Passing Arguments to a Subtask](#)).

In the case of an Item Argument block, the name of the item to be passed is specified in the Argument to Pass attribute of the block



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-arg-to-pass	text		““ opac-local-integer-parameter opac-local-item opac-local-float-parameter

Debugging Palette

Describes the blocks in the Debugging palette.

Introduction **137**

Show Stack Top **138**

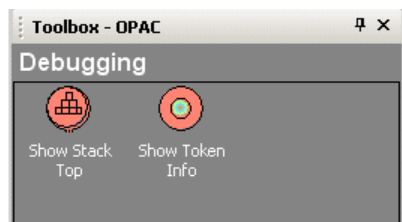
Show Token Info **139**



Introduction

The Debugging palette blocks provide blocks for debugging an Integrity application.

Here is the Debugging palette:



Use these blocks to display Token information when testing and debugging OPAC procedures. You can also use the Block Pause Capability block to view the animated token.

Show Stack Top



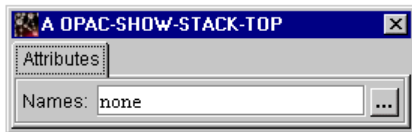
The Show Stack Top block displays the top of the token stack. This block is useful for debugging purposes.

This block is an instance of a General Procedure block. Use this procedure for testing, because in normal operation a G2 window might not be displayed, in which case the procedure picks a window.

During debugging, when you launch a procedure manually from a window, the output is sent to the correct window.

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
N/A			

Show Token Info



The Show Token Info block displays information about an OPAC token. The information provided includes:

- A description of \$block, \$caller, \$target, \$window, \$notify.
- The current value of local parameters attached to the Subtask Start block.

This block is used mainly for debugging.

During debugging, when you launch a procedure manually from a window, the output is sent to the correct window.

When called from a G2 procedure, the information is sent to the SMH Browse Server regardless of \$notify argument specification.

The Show Token Info Procedure is a sample G2 procedure that can be called as the OPAC G2 Action Procedure of a terminal OPAC block. This sample shows how to access the **caller** and **target** for a token. The only argument is the OPAC Token. This is the procedure called by the Show Token Info block:

As with all G2 Action Procedures called from a block, it has a single argument, the Token, and returns an error symbol and error text. It is provided here as an example, so you can see what relations and other functions are available.

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
N/A			

State Transition Palette

Describes the blocks in the State Transition palette.

Introduction	141
State Transition Diagrams	143
Delete State Token	148
Get State	149
Accept New Event	151
Accept New State	153
State Diagram Completion	154
Transition Event	155
Wait State	156
State Diagram Start	157



Introduction

Use the State Transition blocks for creating state transition diagrams. Two groups of blocks are represented on this palette: those that are used to construct a state diagram and those that are used to manipulate a state diagram from within an OPAC procedure.

Here is State Transition palette:



The blocks used to construct a state diagram are:

- [State Diagram Start](#)
- [Wait State](#)
- [Transition Event](#)
- Timeout Transition Event
- [State Diagram Completion](#)

The blocks used to manipulate a state diagram are:

- [Accept New State](#)
- [Accept New Event](#)
- [Get State](#)
- [Delete State Token](#)

This chapter also provides a general description of creating [state transition diagrams](#).

State Transition Diagrams

The Wait State block along with the Transition Event and Timeout Transition Event blocks can be used to build a State Transition model. Each state transition diagram must exist on a separate workspace. A state diagram is constructed by connecting Wait State blocks and Transition Event blocks and/or Timeout Transition Event blocks. Transition blocks must only have one input and one output connection.

A special Start block, State Diagram Start, is used to signify the entry point for a state transition diagram. The first Wait State block connected downstream from the Start block is designated as the initial (default) state of the state transition model. The State Diagram Start block can accept an initial state for the model by passing the state as a string through the `arg-list` parameter of the `opac-start-task` API. A state diagram must be initiated by an OPAC procedure.

The current state of a state transition model is represented by an `opac-token`. The `opac-token` is associated with a Wait State block when the model is not in a transition. The `opac-token` moves from one `opac-wait-state` to an `opac-transition-event` to another `opac-wait-state` when a new event for the model is accepted. Action procedures may be specified both on the `opac-wait-state` and `opac-transition-event` blocks. These action procedures will be executed when the `opac-token` is associated with the block. The procedures may be either an OPAC procedure or a G2 procedure. If the specified procedure names both an OPAC procedure and a G2 procedure the OPAC procedure will be executed.

In order to manipulate a state diagram through OPAC, four OPAC blocks were created: Accept New State block, Accept New Event block, Get State block, and Delete State Token block. These blocks are discussed below.

Two APIs can also be utilized to change the state of a diagram: `accept-event` and `accept-new-state`. Both APIs accept an `opfo-domain-object` and the event name. The difference between the two APIs is the `opac-accept-new-state` will bypass the Transition Event, thus bypassing the transitions action procedure.

Examples of two state diagrams can be viewed and executed by loading in the OPACDEMO.KB module and selecting the 'OPAC State Diagram Examples' button.

Examples

Example State Diagram

Start Example

TEST-STATE-DIAGRAM

opac-std-test

"booting"

Open up the "Example State Diagram". The button labeled "Set Initial State" starts the test-state-diagram OPAC procedure. This procedure calls `opac-std-test` passing one argument, which is the current state for the target, `state-test`. Once the initial state is defined, you can enter a new event and select the "Set New Event" button to transition from one state to another in the diagram. You can set a new state by entering a new state in the type-in-box and select "Set New State". To get the current state for the target, select "Get Current State".

The button below, OPAC State Transition Actions, contains OPAC procedures that utilize blocks that can change the state, send a new event, and retrieve the current state of a given state transition diagram.

Link Down Example

Start Link Down Example

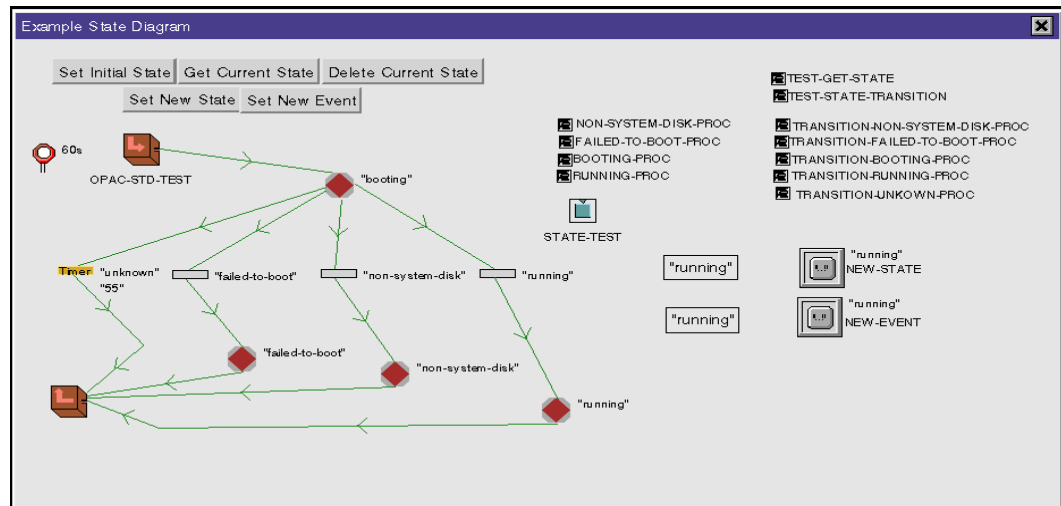
LINK-DOWN-OPAC

link-down-std

OPAC State Transition Actions

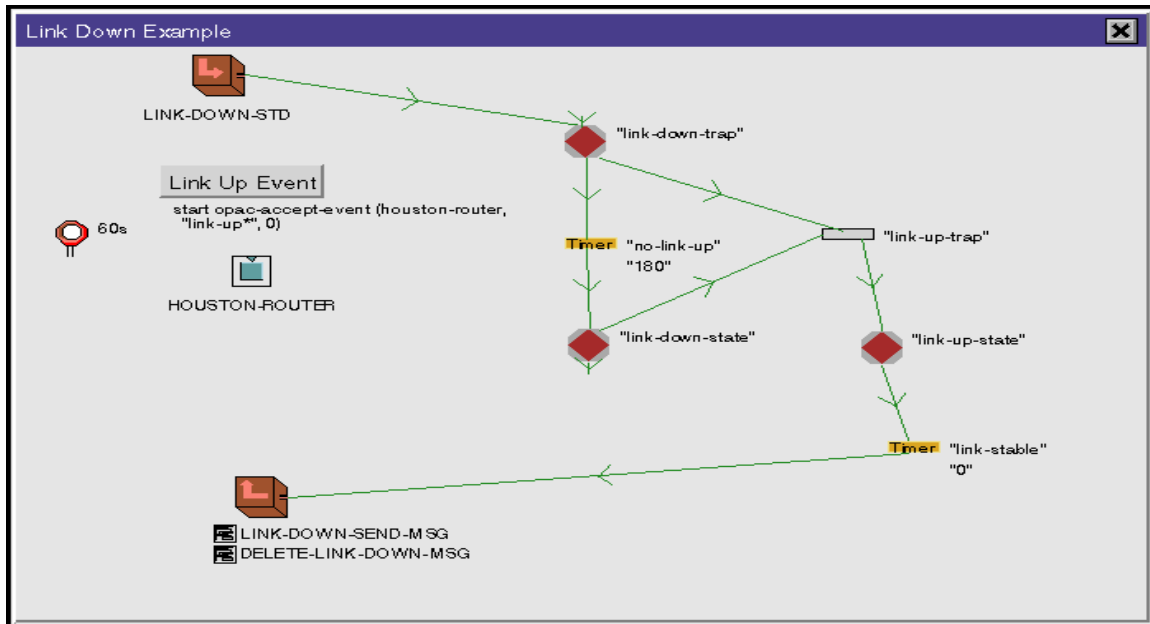
The first example can be viewed by selecting the 'Example State Diagram' button from the Examples workspace. This is a state diagram for the status of a computer

going through a boot process. There are type-in-boxes and buttons to manipulate the state for the diagram.



The System Start example has four states, three transition events, and one timeout transition event. To initiate the state diagram select the 'Start Example' action button on the previous workspace. This will start the test-state-diagram OPAC procedure. The opac-token can be seen waiting at the "booting" state. To move the token to a different state, select either the 'Set New State' or 'Set New Event' buttons. Experimentation can be accomplished by setting the parameter used for the new state (*new-state*) or for the new event (*new-event*) by editing the type-in-boxes next to the parameters. Since this example does have a timeout transition event with a timeout of 55 seconds, it will transition to the completion block if no new state or new event is sent within the 55 seconds.

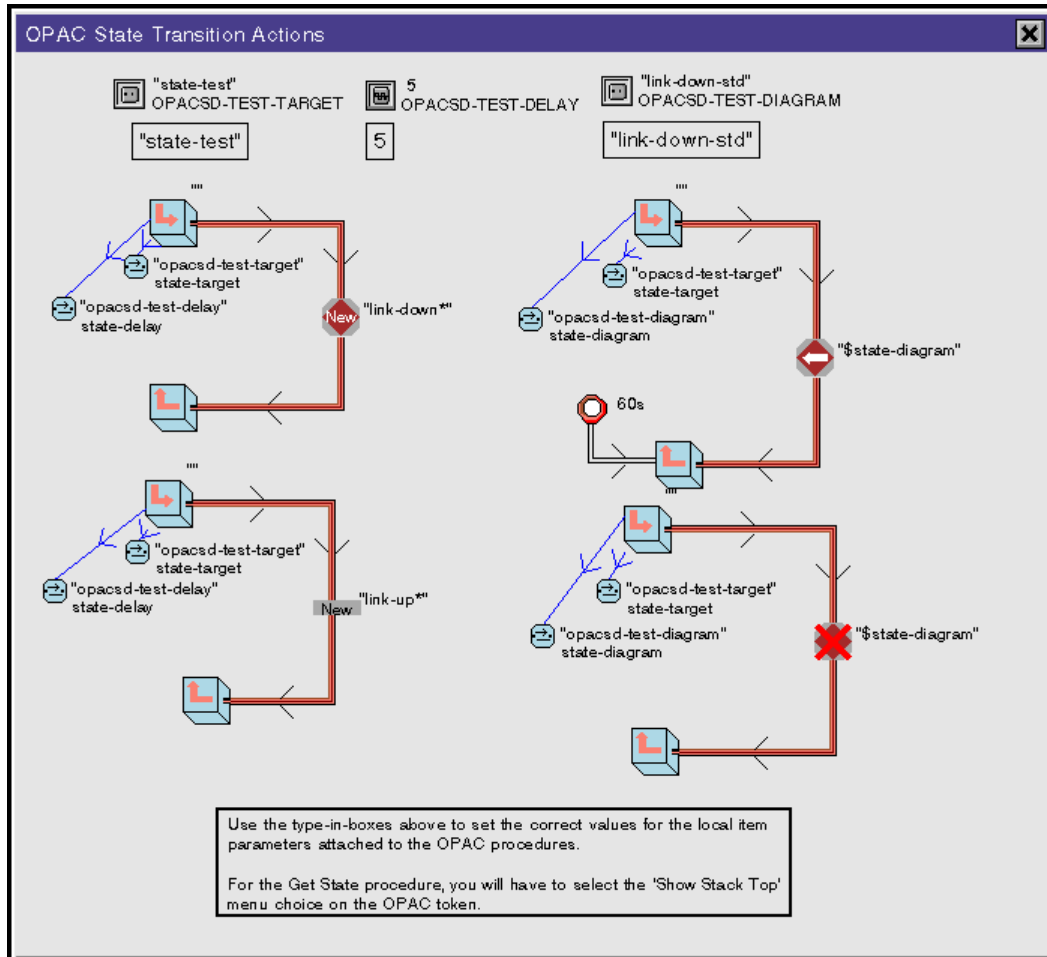
The second example demonstrates a Link Down state diagram.



This example can be viewed by selecting the 'Link Down Example' button from the Examples workspace. This diagram contains three states, one transition event, and two timeout transition events.

To start the Link Down example, select the 'Start Link Down Example' button from the 'Examples' workspace. This will start the link-down-opac procedure. The token can be seen waiting at the "link-down-trap" state. If a "link-up-trap" event is not seen within 180 seconds, the token will transition through the "no-link-up" timeout transition event to the "link-down-state". The action procedure for the "link-down-state" block will generate a message against houston-router. The button, 'Link Up Event', uses the opac-accept-event API and can be used to send a link-up event against the houston-router.

These two examples can also be manipulated through OPAC. Selecting the State Transition Actions button on the Examples workspace shows some simple OPAC procedures which accomplish this.



On this workspace there is an example for each block that can be used to manipulate a state diagram. At the top of the workspace are three parameters that are used as parameters for the OPAC procedures: `opacsd-test-target`, `opacsd-test-delay`, and `opacsd-test-diagram`. The `opacsd-test-target` identifies the Target, which can be any `opfo-domain-object`. The `opacsd-test-delay` is an integer value used in the new event OPAC procedure. The `opacsd-test-diagram` names the `opac-state-diagram-start` object that is used in the delete state token OPAC procedure. By using the type-in-boxes and by changing the attribute values of these blocks, you can manipulate the example state diagrams.

Delete State Token



The Delete State Token block deletes the token associated with the specified target within the specified state diagram.

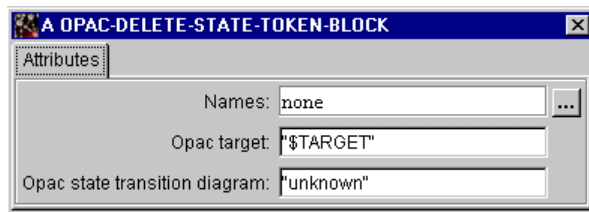
The Target attribute names the target for the new event.

The State Transition Diagram attribute specifies the name of the State Diagram Start block of the state diagram containing the target.

an opac-delete-state-token-block	
Names	none
Opac target	"\$TARGET"
Opac state transition diagram	"unknown"

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-target	text	"\$TARGET"	Any text representing a Target. Local parameters, preceded by a '\$' may also be used
opac-state-transition-diagram	text	"unknown"	The name of any opac-state-diagram-start block

Get State



The Get State block retrieves the current state for the specified target within the specified state transition diagram and places the result at a specified destination.

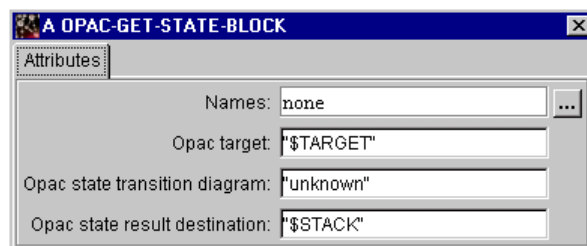
The block uses three attributes:

- The Target attribute names the target from which to retrieve the state.
- The State Transition Diagram attribute contains the name of the state diagram from which to retrieve the state. A target may have several state diagrams active simultaneously.
- The State Result Destination represents the location at which to place the retrieved state.

an opac-get-state-block	
Names	none
Opac target	"\$TARGET"
Opac state transition diagram	"unknown"
Opac state result destination	"\$STACK"

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-target	text	“\$TARGET”	Any text representing a Target. Local parameters, preceded by a ‘\$’ may also be used
opac-state-transition-diagram	text	“unknown”	Name of any opac-state-diagram-start block
opac-state-result-destination	text	“\$STACK”	Location to place the results of the query

Accept New Event



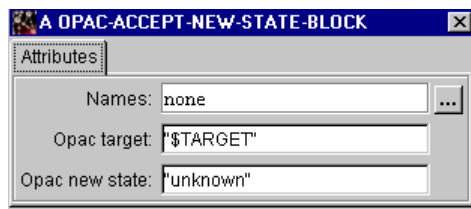
The Accept New Event block is used to move the opac-token to a new state based on the specified event. Three attributes are used:

- The Target names the target for the new event.
- The New Event contains the new event for the target.
- The Event Delay value represents the number of seconds by which to delay the propagation of the new event.

an opac-accept-new-event-block	
Names	none
Opac target	"\$TARGET"
Opac new event	"unknown"
Opac event delay	"0"

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-target	text	"\$TARGET"	Any text representing a Target. Local parameters, preceded by a '\$' may also be used

Attribute	Data Type	Default Value	Possible Values
opac-new-event	text	“unknown”	Any user-defined state
opac-event-delay	text	“0”	Number of seconds to delay the new event

Accept New State

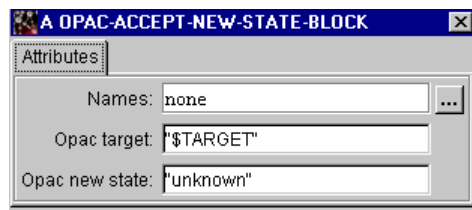


The Accept New State block moves the token to a new state, bypassing any transition event blocks. The Target attribute names the target for the new state. The New State attribute specifies the new state. Because this block bypasses any transition event blocks, the transition event's action procedure is not executed.

an opac-accept-new-state-block	
Names	none
Opac target	"\$TARGET"
Opac new state	"unknown"

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-target	text	"\$TARGET"	Any text representing a Target. Local parameters, preceded by a '\$' may also be used
opac-new-state	text	"unknown"	Any user-defined state

State Diagram Completion



The State Diagram Completion block indicates the end of processing designated by a State Transition diagram.



Transition Event



The Transition Event block waits for the specified transition event and then executes the specified Event Action Procedure. Transition Event blocks are connected to no more than one Wait State block. The method or procedure can be either a G2 procedure or an OPAC procedure.

Properties Dialog

The Properties dialog for the block is shown below:

Attributes

Attribute	Data Type	Default Value	Possible Values
opac-transition-event-name	text	“unknown”	Any user-defined transition event name.
opac-event-action-proc	text	“default”	Default, Opac procedure name, or any user-defined procedure or method.

Wait State



The Wait State block defines a system state and its associated procedure. Specify the system state in the State attribute. Specify the associated procedure in the Wait State Action Procedure attribute.

Properties Dialog

The Properties dialog for the block is shown below:

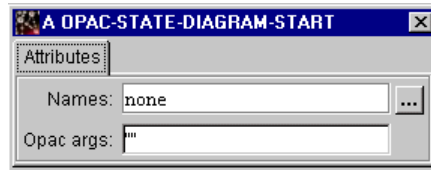
Attributes

Attribute	Data Type	Default Value	Possible Values
opac-state	text	"unknown"	Any user-defined state.
opac-wait-state-action-proc	text	"default"	Default, name of an OPAC procedure, or any user-defined procedure or method.

State Diagram Start



The State Diagram Start block indicates the start of processing designated by a State Transition diagram. Specify arguments to the state diagram process by specifying the Args attribute.



External Interfaces Palette

Describes the blocks in the External Interfaces palette.

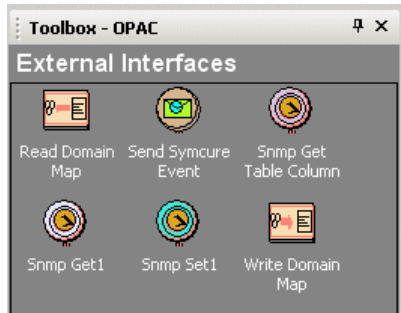
Introduction	160
Read Domain Map	161
Write Domain Map	164
SNMP Get Table Column	166
SNMP Set	168
SNMP Get	170
Send CDG Event	172



Introduction

This chapter describes the External Interfaces blocks. This palette contains blocks used to interface with external systems. This includes SymCure, formerly known as CDG, Operations Expert SNMP (OXS), and blocks used to read and write domain maps.

Here is the External Interfaces palette:



Read Domain Map

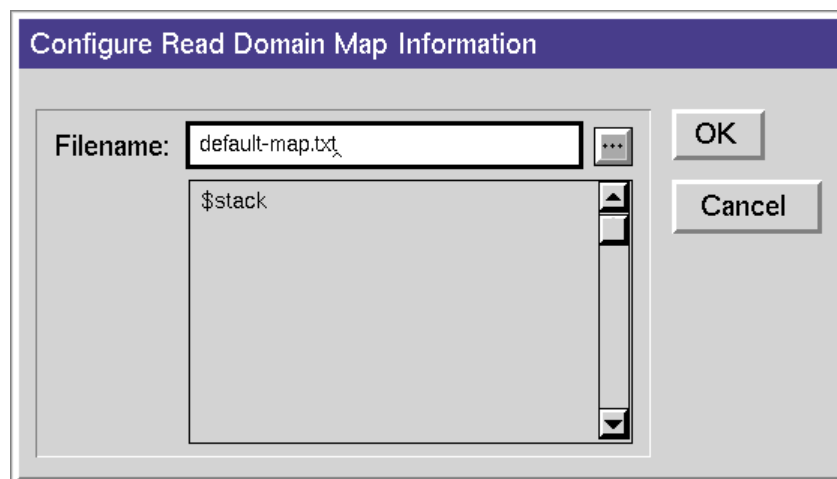


The Read Domain Map block reads an ASCII text file and creates domain objects and connections based on the information contained in the file. Refer to the *Integrity User's Guide* for file format. You must specify:

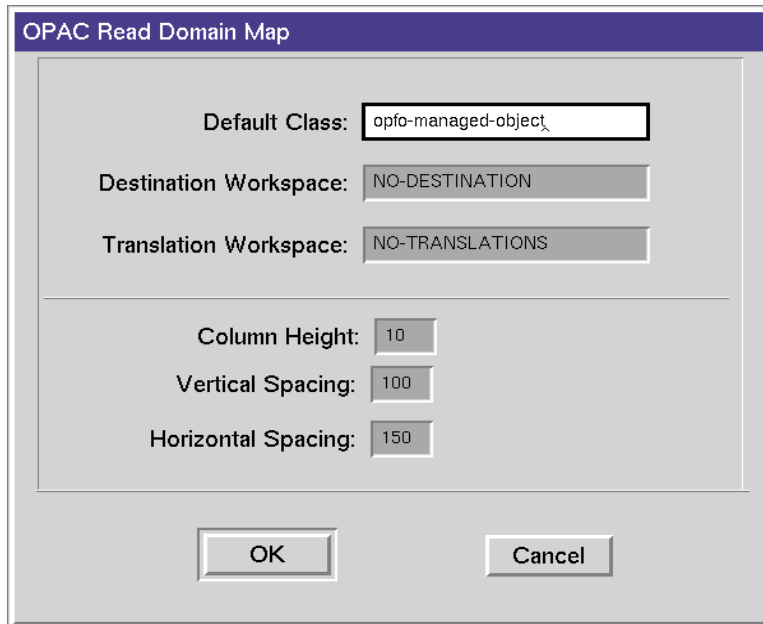
- Default class
- Destination workspace
- Translation workspace
- Column height
- Vertical and horizontal spacing

The default class must be a subclass of `opfo-domain-object`. The destination workspace is the location where the resulting map is placed. The translation workspace is a workspace that contains oid-to-name translation objects. These are used to convert OID numbers to a text description for easy recognition. The column height and the vertical and horizontal spacing are values for laying out the objects for the map.

This block is an interface into the Import Map option on the File menu bar. Use this to update or build a current domain map.



When you click OK, the following dialog appears:



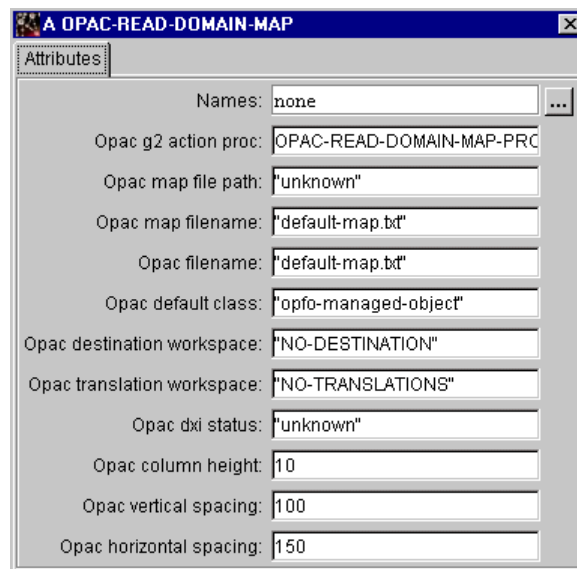
Attributes

Attribute	Data Type	Default Value	Possible Values
opac-default-class	text	"opfo-managed-object"	Any valid opfo-domain-object or subclass of opfo-domain-object.
opac-destination-workspace	text	"NO-DESTINATION"	Any workspace, subworkspace, or object. A subworkspace will be created for objects which do not have subworkspaces.
opac-translation-workspace	text	"NO-TRANSLATIONS"	Any workspace containing translation objects.
opac-column-height	integer	10	Any positive value.

Attribute	Data Type	Default Value	Possible Values
opac-vertical-spacing	integer	100	Any positive value.
opac-horizontal-spacing	integer	150	Any positive value.
opac-filename	text	"default=map.txt"	Any valid filename and path

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-filename	text	"default-map.txt"	Any valid filename and path.

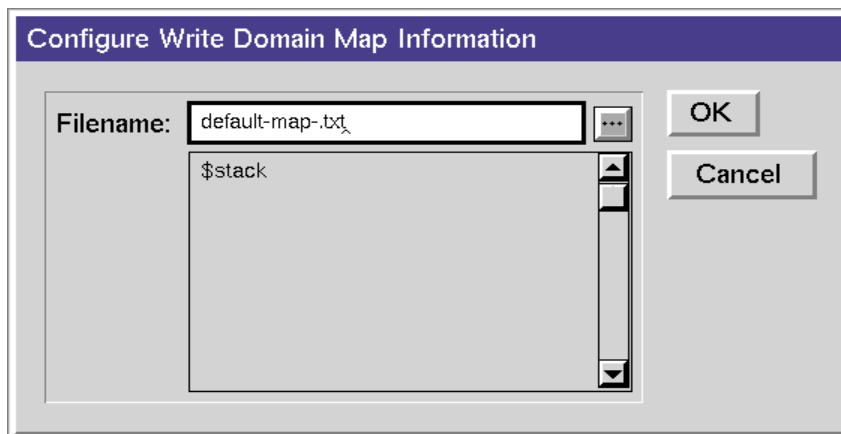
Write Domain Map



The Write Domain Map block creates an ASCII text file from the top-level object specified in the Top Object attribute. Specify the file name that the domain map will be written to in the Filename attribute. Refer to the *Integrity User's Guide* for the file format.

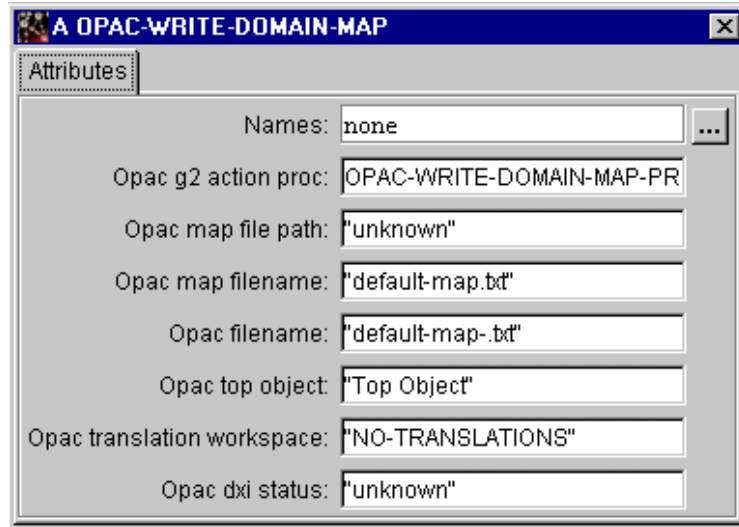
The user-specified top-level object is the object used as the starting point for writing the map. The translations workspace can be the same translation workspace as described in the Read Domain Map section.

This block is an interface to the Export Map menu choice of the File menu. Use this block to write a file containing containment information about the domain map.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-filename	text	"default-map.txt"	Any valid filename or path.
opac-top-object	text	"Top Object"	Any valid object.
opac-translation-workspace	text	"NO-TRANSLATIONS"	Any workspace containing translation objects.

SNMP Get Table Column



The SNMP Get Table Column block retrieves a table of values for a specified Object ID from the target specified in the Agent Hostname attribute.

The procedure has execution time limit capability, specified by the Timeout (seconds) attribute. If the result is not returned within this limit, the procedure aborts.

Configure SNMP Get Table Column

ASN1 Type: 4

Data Source: Live Simulated

Agent Hostname: \$target

Object ID: "1.3.6.1.2.1.1.1.0"

Results Destination: "\$stack"

Community: public

Timeout (seconds): 30

OK

Cancel

\$stack

\$caller

Properties Dialog

The Properties dialog for the block is shown below:

A OPAC-SNMP-GET-TABLE-COLUMN

Attributes

Names: none

Opac oid: "1.3.6.1.2.1.1.1.0"

Opac asn1 type: 4

Opac agent hostname: "\$target"

Opac community: "public"

Opac timeout: 30

Opac snmp type: SNMP

Opac snmp interface: SNMP-SYNCHRONOUS-INTERI

Opac results destination: "\$stack"

Attributes

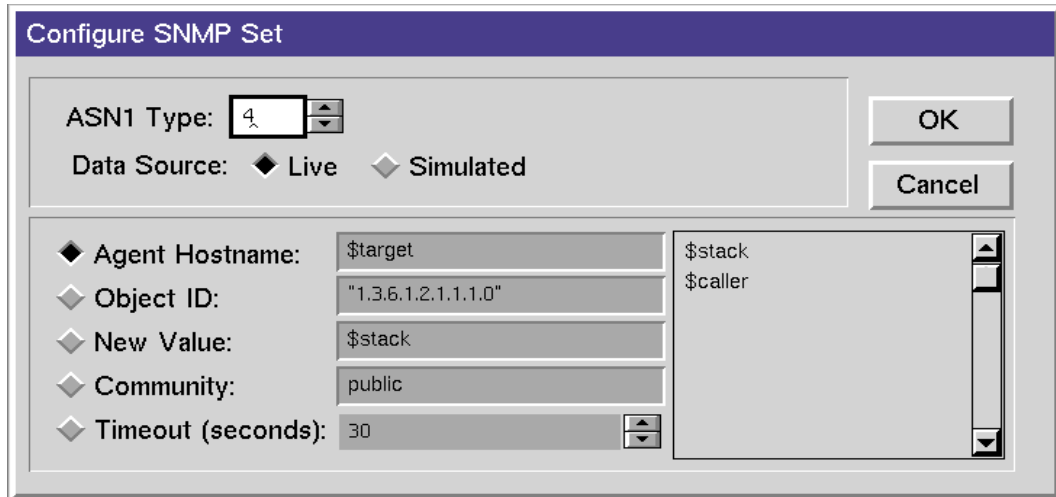
Attribute	Data Type	Default Value	Possible Values
opac-asn1-type	integer	4	4 = octet string 2 = integer
opac-snmp-type	symbol	snmp	Live (snmp) or simulated (simulated-snmp).
opac-agent-hostname	text	“\$target”	Any hostname. (\$) references will be resolved.
opac-oid	text	“1.3.6.1.2.1.1.1.0”	Any number representing an OID. (\$) references will be resolved.
opac-results-destination	symbol	\$stack	\$stack or an opac-local-item.
opac-community	text	“public”	Any community string. (\$) references will be resolved.
opac-timeout	number	30	Any number. (\$) references will be resolved.

SNMP Set



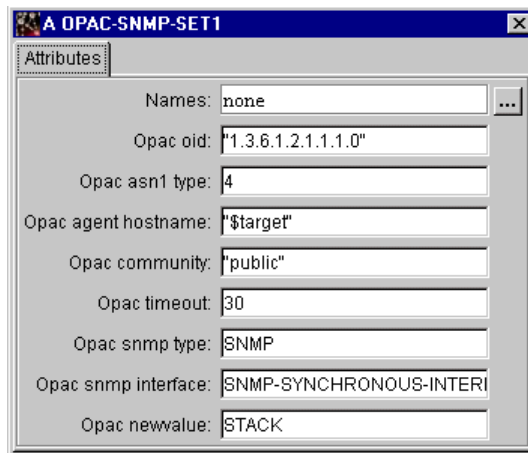
The SNMP Set block performs an SNMP Set request to set a value of a single variable.

The procedure has execution time limit capability, specified by the Timeout (seconds) attribute. If the result is not returned within this limit, the procedure aborts.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-asn1-type	integer	4	4 = octet string 2 = integer
opac-snmp-type	symbol	snmp	Live (snmp) or simulated (simulated-snmp).
opac-agent-hostname	text	“\$target”	Any hostname. (\$) references will be resolved.
opac-oid	text	“1.3.6.1.2.1.1.1.0”	Any number representing an OID. (\$) references will be resolved.
opac-new-value	symbol	\$stack	Same as Get.
opac-community	text	“public”	Any community string. (\$) references will be resolved.
opac-timeout	integer	30	Any number. (\$) references will be resolved.

SNMP Get



The SNMP Get block performs an SNMP Get request, to get the value of a single variable.

The procedure has execution time limit capability, specified by the Timeout (seconds) attribute. If the result is not returned within this limit, the procedure aborts.

The 'Configure SNMP Get' dialog box contains the following fields and options:

- ASN1 Type: 4
- Data Source: Live Simulated
- Agent Hostname: \$target
- Object ID: "1.3.6.1.2.1.1.1.0"
- Results Destination: \$stack
- Community: public
- Timeout (seconds): 30

Buttons: OK, Cancel

Properties Dialog

The Properties dialog for the block is shown below:

The 'A OPAC-SNMP-GET1' Properties dialog box shows the following attributes:

- Names: none
- Opac oid: "1.3.6.1.2.1.1.1.0"
- Opac asn1 type: 4
- Opac agent hostname: "\$target"
- Opac community: "public"
- Opac timeout: 30
- Opac snmp type: SNMP
- Opac snmp interface: SNMP-SYNCHRONOUS-INTERI
- Opac results destination: "\$stack"

Attributes

Attribute	Data Type	Default Value	Possible Values
opac-asn1-type	integer	4	4 = octet string 2 = integer
opac-snmp-type	symbol	snmp	Live (snmp) or simulated (simulated-snmp).
opac-agent-hostname	text	“\$target”	Any hostname, such as a domain object. (\$) references will be resolved.
opac-oid	text	“1.3.6.1.2.1.1.1.0”	Any number representing an OID. (\$) references will be resolved.
opac-results-destination	symbol	\$stack	Stack, \$stack, an opac-local-integer-parameter, or an opac-local-text-parameter.
opac-community	text	“public”	Any community string. (\$) references will be resolved.
opac-timeout	integer	30	Any number. (\$) references will be resolved.

Send CDG Event



The Send CDG Event block sends an event to SymCure for processing. Specify the information necessary to identify the event to SymCure by the following attributes:

- The Target attribute specifies the domain object against which the event occurred.
- The Sender attribute specifies the sender of the event.
- The Category attribute specifies.
- The Event type attribute specifies whether it is a fault, test, or symptom, and the type of fault, test or symptom.
- The Event Value attribute specifies the value associated with the event.

You can also extract this information from the token by using `$target`, the category of `$caller`, if the block is called with appropriate `target` and `caller`.

An event to SymCure is uniquely identified by its:

- Target The domain object against which the event came in.
- Sender The object sending the event.
- Category A unique text identifying the event.
- Event type A fault, symptom, or test.
- Event value True, false, suspect, or unknown.

Use the Send CDG Event block to send an event to SymCure. The block works only when the SymCure module is loaded in Operations Expert.

Configure Send CDG Event

Event type: ▲

Event Value: ▲
 TRUE = 1.0
 FALSE = 0.0

◆ Target: \$stack
 ◆ Sender: \$caller
 ◆ Category:

OK
Cancel

Properties Dialog

The Properties dialog for the block is shown below:

A OPAC-SEND-CDG-EVENT [X]

Attributes

Names: ...

Opac target:

Opac sender:

Opac category:

Opac event type:

Opac event val:

Attributes

Attribute	Data Type	Default Value	Possible Values
opac-event-type	text	“symptom”	Any one of: <ul style="list-style-type: none">• generic-fault• generic-or-test• generic-and-test• generic-or-symptom• generic-and-symptom
opac-event-val	float	0.0	Any fuzzy truth-value between 0.0 (False) and 1.0 (True)
opac-target	text	“\$target”	The domain object against which the event came in.
opac-sender	text	“\$caller”	The object sending the event.
opac-category	text	“the cdg-event-name of \$caller”	Any (\$) reference will be resolved.

Generic Blocks Palette

Describes the blocks in the Generic Blocks palette.

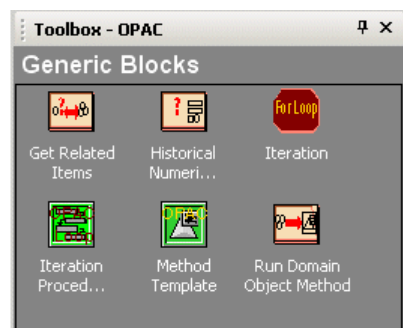
- Introduction **175**
- Get Related Items **176**
- Historical Numerical Query **178**
- Iteration **180**
- Run Domain Object Method **182**
- Set Local Parameter From Source **184**



Introduction

Use the Generic Blocks for general applications development along with G2 Action Procedures.

Here is the Generic Blocks palette:



Get Related Items



The Get Related Items block collects items that are related to a specified object by a G2 or user-defined relation. The collected items are placed in a list on the token stack. Specify the focus object in the Referenced Object attribute.

Specify the relationship of interest in the Relationship attribute. Valid relations include *connected-to*, *superior-to*, *subordinate-to*, *connected-upstream-to*, *connected-downstream-to*, or any user-defined relation.

A class filter can be used to limit the items collected to those objects within a specified class. Specify the class by which to filter in the Domain Class Filter attribute.

The screenshot shows a dialog box titled "OPAC Get Related Items Block". It contains the following fields and controls:

- Referenced Object:** A text field containing "\$target".
- Object List:** A list box containing "\$stack" and "\$caller".
- Relationship:** A text field containing "connected-to".
- Domain Class Filter:** A text field containing "item".
- Buttons:** "OK" and "Cancel" buttons at the bottom.

Properties Dialog

The Properties dialog for the block is shown below:

The screenshot shows a Properties dialog box titled "A OPAC-GET-RELATED-ITEMS-BLOCK". It contains the following fields:

- Names:** A text field containing "none".
- Opac referenced obj:** A text field containing "\$target".
- Opac relationship:** A text field containing "connected-to".
- Opac domain class filter:** A text field containing "item".

Attributes

Attribute	Data Type	Default Value	Possible Values
opac-referenced-obj	text	“\$target”	Any subclass of opfo-managed-object or opfo-containment-object
opac-relationship	text	“connected-to”	connected-to, superior-to, subordinate-to, connected-upstream-to, connected-downstream-to, or any user defined relation
opac-domain-class-filter	text	“item”	Any valid class to search on.

Historical Numerical Query



The Historical Numerical Query block detects and generates events based on numerical variables. Typically, these numerical values are from performance monitoring (statistical quality control). The block allows access to and calculation of statistics based on historical numerical data stored in or delegated by domain objects.

- Specify the target object in the Target attribute.
- Specify an attribute of the target object in the Attribute Name attribute.
- Specify the statistical function to perform, such as **average**, **minimum**, **maximum**, **rate-of-change**, **integral**, or **derivative**, in the Statistical Function attribute.
- Specify the time period (in minutes) over which to perform the calculation in the Query Time Interval (Minutes) attribute.

A numerical result is returned to the token stack. This might then be compared to a threshold, for instance.

You must supply methods for the domain objects, which may directly contain the numerical attributes, or use any calculation method, or delegate the query to some external source of information, such as a database.

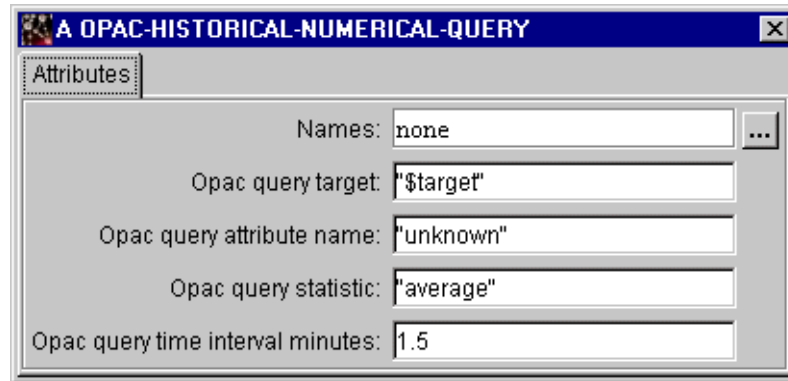
The screenshot shows a dialog box titled "OPAC Historical Numerical Query". It contains several input fields and a list box:

- Target:** A text field containing "\$target". Below it is a list box with the items "\$stack" and "\$caller".
- Attribute Name:** A text field containing "unknown".
- Statistical Function:** A text field containing "average".
- Query Time Interval (Minutes):** A text field containing "1.5".

At the bottom of the dialog are two buttons: "OK" and "Cancel".

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-query-target	text	“\$TARGET”	Any subclass of opfo-domain-object or opfo-containment-object
opac-query-attribute-name	text	“unknown”	Any valid attribute of the message object.
opac-query-statistic	text	“average”	average maximum minimum interpolated rate-of-change standard-deviation value-of
opac-query-time-interval-minutes	integer	“1.5”	Any valid interval time in minutes.

Iteration



The Iteration block operates similarly to a for loop and allows you to specify configuration of three possible iterations:

- over all members of a class
- over a list of objects
- over a range of numbers

To iterate over all members of a class, specify the word, “class” in the Iteration Type attribute, specify the class in the Iteration Class attribute, and specify the method to use for iteration in the Method attribute.

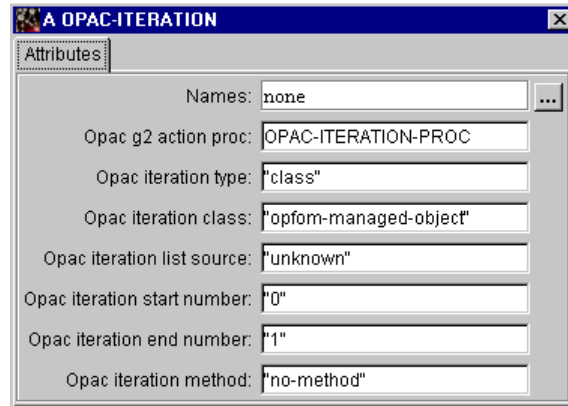
To iterate over a list of objects, specify the word, “list” in the Iteration Type attribute, specify the variable (local to the OPAC procedure or a permanent list object) in the List Source attribute, and specify the procedure, method, or OPAC procedure to use for iteration in the Method attribute.

To iterate over a range of numbers, specify the starting integer value in the Start Number attribute, specify the ending integer value in the End Number attribute, and specify and the procedure, method, or OPAC procedure to use for iteration in the Method attribute.

A screenshot of a dialog box titled "OPAC Iteration". The dialog box has a purple header bar with the title in white. Below the header, there are several labeled input fields: "Iteration Type:" with a text box containing "class"; "Iteration Class:" with a text box containing "opfom-mana ..."; "List Source:" with a text box containing "unknown"; "Start Number:" with a text box containing "0"; "End Number:" with a text box containing "1"; and "Method:" with a text box containing "no-method". At the bottom of the dialog box, there are two buttons: "OK" and "Cancel".

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-iteration-type	text	"class"	"class", "list", or "range"
opac-iteration-class	text	"opfom-managed-object"	Any subclass of opfo-managed-object or opfo-containment-object.
opac-iteration-list-source	text	"unkown"	Any list reference.
opac-iteration-start-number	text	"0"	Any valid integer representing a starting count
opac-iteration-end-number	text	"1"	Any valid integer representing an ending count
opac-iteration-method	text	"no-method"	Any user-defined method which operates on the opac-iteration-class

Run Domain Object Method



The Run Domain Object Method block allows you to specify running a specified method against a specified object. You can configure the block to place a specified type of return value on the stack of the token. You can also specify a class filter.

Specify the object in the Referenced Object attribute.

Specify the method to run against the object in the Domain Method attribute.

You can filter the objects to those of a specified class. To limit the referenced objects to a specific class, specify the class in the Domain Class Filter attribute.

Specify the return type in the Domain Method Return type attribute. Valid return types include none, object, truth-value, or list.

OPAC Run Domain Object Method

◆ Referenced Object:

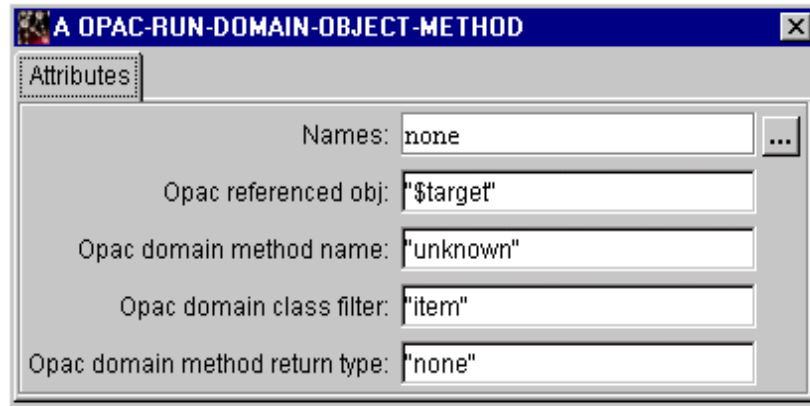
Domain Method:

Domain Class Filter:

Domain Method Return Type:

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

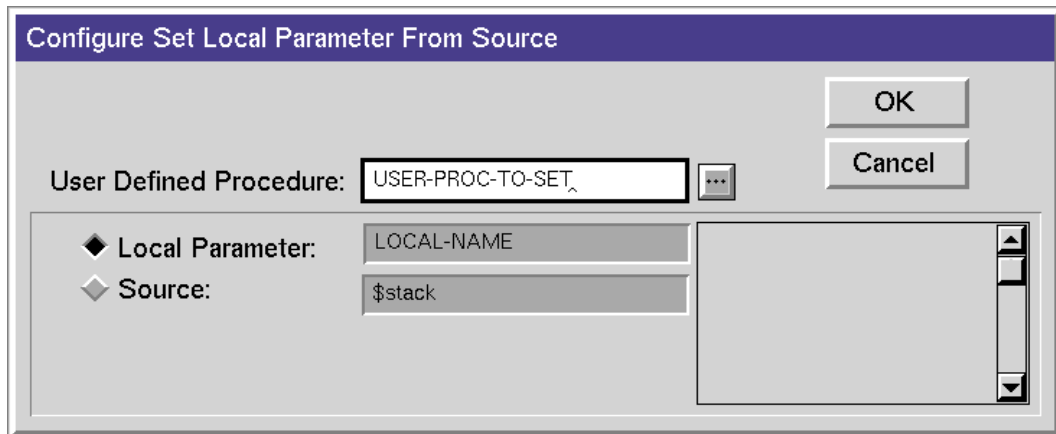
Attribute	Data Type	Default Value	Possible Values
opac-referenced-obj	text	“\$target”	Any subclass of opfo-managed-object or opfo-containment-object
opac-domain-method-name	text	“unknown”	Any valid method for the class specified by opac-referenced-obj
opac-domain-class-filter	text	“item”	Any class to search on.
opac-domain-method-return-type	text	“none”	none object truth-value item-list

Set Local Parameter From Source



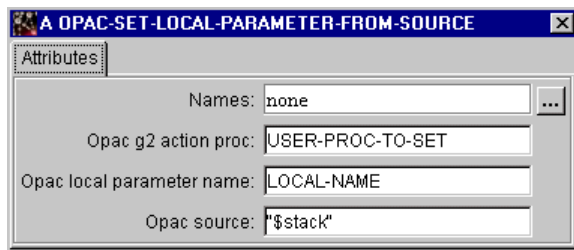
The Set Local Parameter From Source block allows you to use a user-defined G2 procedure to assign a value to a local parameter.

Use this block only with an existing user-defined G2 procedure. Specify the procedure name in the User Defined Procedure attribute of the Configure dialog box or the OPAC G2 Action Procedure attribute of the Properties dialog for this block. Specify the local parameter that will be set by the procedure in the OPAC Local Parameter Name attribute.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-local-parameter	symbol	local-name	Any local parameter name.

Message Palette

Describes the blocks in the Message palette.

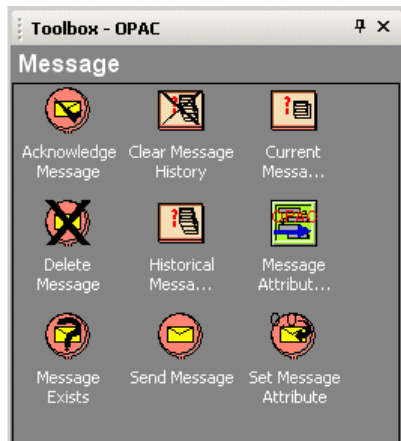
Introduction	185
Send Message	187
Current Message Query	190
Clear Message History	193
Message Exists	195
Set Message Attribute	197
Delete Message	200
Acknowledge Message	202



Introduction

Use the Message blocks for general message manipulation including acknowledging a message, deleting a message, setting a message attribute, testing for the existence of a message, clearing the message history, querying the current message, and sending a message.

Here is the Message palette:



Send Message



The Send Message block creates and sends a message to the specified message server. The block sends messages only to servers in the same G2 process.

The attributes of the block match the calling arguments for the SMH Create Message block, the procedure API for creating and sending any message.

For information “Message Handling” refer to the *Integrity User’s Guide*.

Configure Send SMH Message

Message Text:

Additional Text:

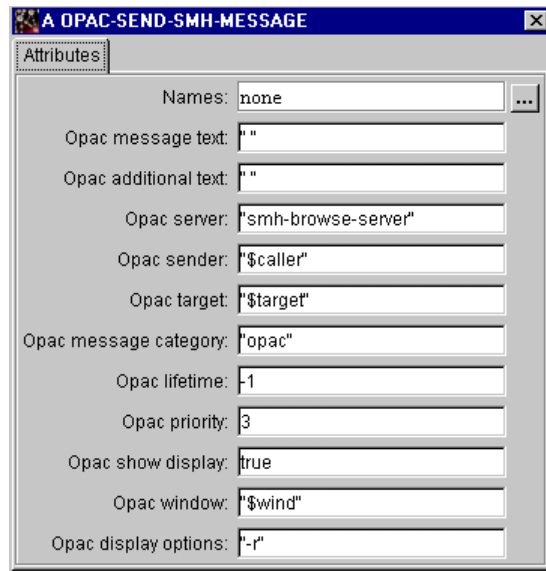
◆ Server:	<input style="width: 95%;" type="text" value="smh-error-server"/>	mjl-message-server
◆ Target:	<input style="width: 95%;" type="text" value="\$target"/>	
◆ Sender:	<input style="width: 95%;" type="text" value="\$caller"/>	
◆ Category:	<input style="width: 95%;" type="text" value="opac"/>	
◆ Window:	<input style="width: 95%;" type="text" value="\$wind"/>	
◆ Priority:	<input style="width: 95%;" type="text" value="1"/>	

Display Options:
 append
 ignore duplicate
 replace
 acknowledgement
 timestamp history

Lifetime:
 weeks:
 days:
 hours:
 minutes:
 seconds:

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-message-text	text	""	Any text of message. If the value contains (\$) references, they will be resolved.
opac-additional-text	text	""	Any text. If the value contains (\$) references, they will be resolved.
opac-server	text	"smh-browse-server"	Any valid message server
opac-sender	text	"\$caller"	Any object
opac-target	text	"\$target"	Any object

Attribute	Data Type	Default Value	Possible Values
opac-message-category	text	"opac"	any user-defined category. If the value contains (\$) references, they will be resolved.
opac-lifetime	number	-1	Lifetime of message in seconds. Any number < 0 means infinite lifetime.
opac-priority	number	3	Any priority (1 - 10). Local integer parameters can also be used as they will be resolved.
opac-show-display	symbol	true	True or false.
opac-window	text	"\$wind"	A g2-window or object.
opac-display-options	text	"-r"	"-nolist" and/or "-nack" and/or "-i" or "-a" or "-r"

Current Message Query



The Current Message Query block returns either a list of current messages or the count of current messages that match the specified criteria. This block differs from the Historical Message Query, because it places the results on the token stack. The object placed on the stack can be either the list of messages that match the query or the count of messages.

To return a list of messages that meet the criteria, specify “list” in the Return Type attribute, or, to return a count of messages that meet the criteria, specify “count” in the Return Type attribute.

To narrow the search, you can specify the following:

- The message target in the Target attribute
- The message sender in the Sender attribute
- The message category in the Category attribute
- A particular server in the Server attribute. Or you can specify “all servers in this process” in the Server attribute.

To search on an attribute, specify the name and value of the attribute in the Attribute and Value attributes, respectively.

OPAC Current Message Query

◆ Target:

◆ Sender:

◆ Category:

◆ Server:

\$stack
\$caller

Category Starting Position: Attribute:

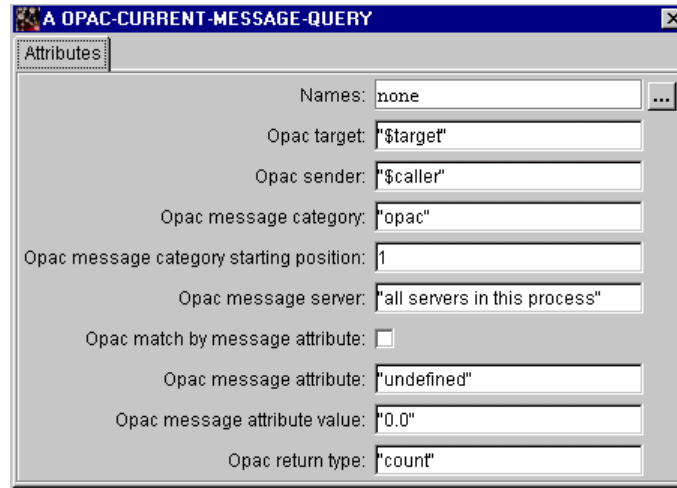
Return Type: Value:

Match by Attribute:

OK Cancel

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-target	text	"\$target"	Any subclass of opfo-managed-object or opfo-containment-object.
opac-sender	text	"\$caller"	Any object.
opac-message-category	text	"opac"	Any user-defined category. If the value contains (\$) references, they will be resolved.
opac-message-server	text	"all servers in this process"	Any valid message server. The default implies all message servers.
opac-message-category-starting-position	integer	1	Any number valid in the range of 1 to the length of the category of message.

Attribute	Data Type	Default Value	Possible Values
opac-return-type	text	“count”	“count” or “list”
opac-message-attribute	text	“undefined”	Any valid attribute of a message.
opac-message-attribute-value	text	“0.0”	Any valid value for the message attribute.
opac-match-by-message-attribute	symbol	false	TRUE or FALSE

Clear Message History



The Clear Message History block clears the smh-histories for specified messages.

To identify the messages, specify the following:

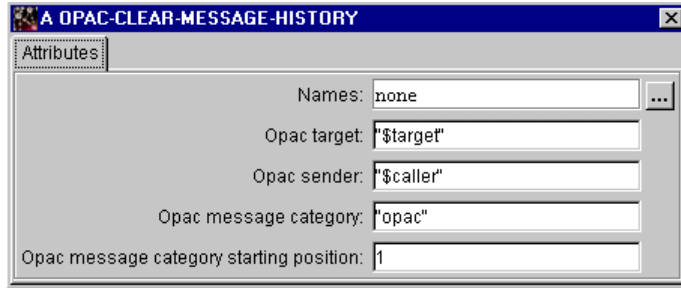
- The message target in the Target attribute
- The message sender in the Sender attribute
- The message category in the Category attribute
- The starting position in the message for the category in the Category Starting Position

Messages are referenced by Target, Sender, and Category. The Target and Sender may be either a single Target and Sender or a list of Targets and Senders. Wild cards may be used in specifying the Category.

The screenshot shows a dialog box titled "OPAC Clear Message History". It contains several input fields and a list box. The "Target:" field contains "\$target_", the "Sender:" field contains "\$caller", and the "Category:" field contains "opac". The "Category Starting Position:" field contains "1". To the right of these fields is a list box containing "\$stack" and "\$caller". At the bottom of the dialog are "OK" and "Cancel" buttons.

Properties Dialog

The Properties dialog for the block is shown below:



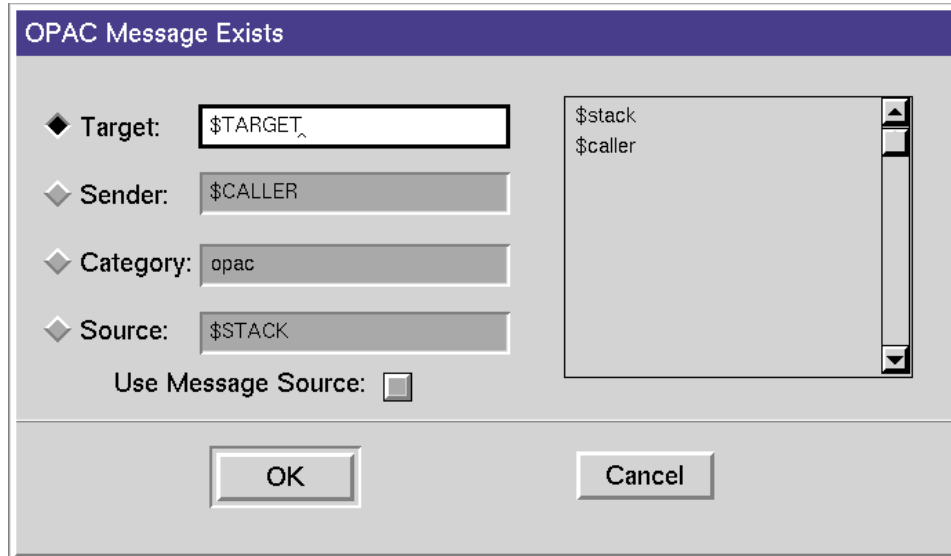
Attributes

Attribute	Data Type	Default Value	Possible Values
opac-target	text	“\$target”	Any subclass of opfo-domain-object or opfo-containment-object.
opac-sender	text	“\$caller”	Any object
opac-message-category	text	“opac”	Any user-defined category. If the value contains (\$) references, they will be resolved.
opac-message-category-starting-position	integer	1	Any number valid in the range of 1 to the length of the category of message.

Message Exists

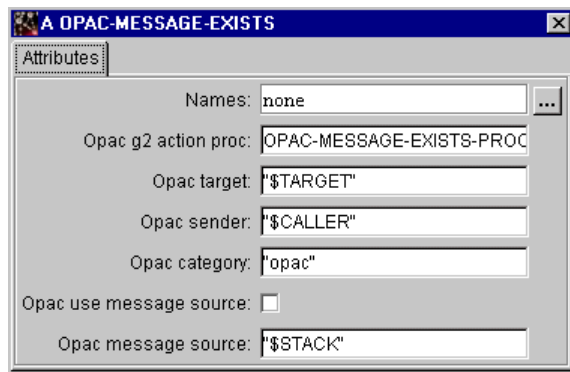


The Message Exists block determines whether a message, identified by the specified target, sender, and category, exists on a server. The block returns a truth-value to the token stack. A **true** value signifies that a message identified by Target, Sender, and Category exists. A **false** value signifies a message does not exist. This block searches all messages servers or a supplied message server. This block does not base its results on any histories of messages.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-target	text	“\$TARGET”	Any subclass of opfo-managed-object or opfo-containment-object.
opac-sender	text	“\$CALLER”	Any object.
opac-category	text	“opac”	Any user-defined category. If the value contains (\$) references, they will be resolved.
opac-message-source	text	“\$STACK”	Any valid reference to a message object.
opac-use-message-source	symbol	FALSE	TRUE or FALSE

Set Message Attribute



The Set Message Attribute block sets the supplied value to the specified message attribute. You can specify the message either by the combination of Target, Sender, and Category, or based on a message object, referenced by a local variable. Local variables are displayed in the list box to the right of the dialog. The variables displayed in the list box are local only to the OPAC procedure to which this block is connected. To obtain the message by the variable in the Source attribute, select the Use Message Source button. Optionally, you can specify a method by which to set the particular attribute. To invoke a user-defined method on the message object, select Use User Method, and specify the method in the User Method attribute.

The user method must include two formal arguments:

- The first argument is the message object.
- The second argument is the value which is set in the dialog.

The method must not return a value.

OPAC Set Message Attribute

◆ Target:

◇ Sender:

◇ Category:

◇ Source:

Use Message Source:

Attribute: Value:

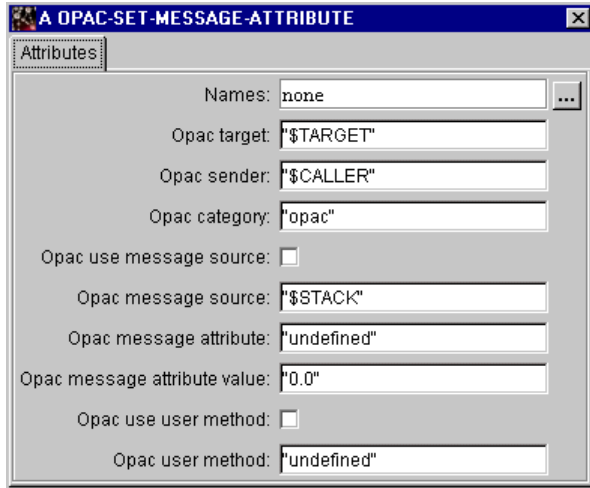
User Method:

Use User Method:

OK Cancel

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-target	text	"\$TARGET"	Any subclass of opfo-managed-object or opfo-containment-object.
opac-sender	text	"\$CALLER"	Any object.
opac-category	text	"opac"	Any user-defined category. If the value contains (\$) references, they will be resolved.
opac-message-source	text	"\$STACK"	Any valid reference to a message object.
opac-use-message-source	symbol	FALSE	TRUE or FALSE.
opac-message-attribute	text	"undefined"	Any valid attribute of a message.

Attribute	Data Type	Default Value	Possible Values
opac-message-attribute-value	text	"0.0"	Any valid value for the attribute.
opac-user-method	text	"undefined"	Any valid user-defined method.
opac-use-user-method	symbol	FALSE	TRUE or FALSE.

Delete Message



The Delete Message block deletes messages specified by target, sender, and category, or by a local item variable, from the message server. The action of the block includes deletion of unacknowledged messages that meet the criteria. Using the Delete Message block, you can delete messages in one of two ways:

- Based on the supplied Target, Sender, and Category.
- Based on a local item variable.

When referencing the message by Target, Sender, and Category, you can use wild cards for the Category.

The local item variable can be either a single message object, or a list containing message objects. The local item variable is connected to the OPAC procedure Start Block. You can also specify taking the reference from the token stack by referencing `$stack` as the Source.

Note A message does not have to be acknowledged to be deleted by the Delete Message block. The block deletes messages that meet the specified criteria that are unacknowledged.

OPAC Delete Message

◆ Target:

◆ Sender:

◆ Category:

◆ Source:

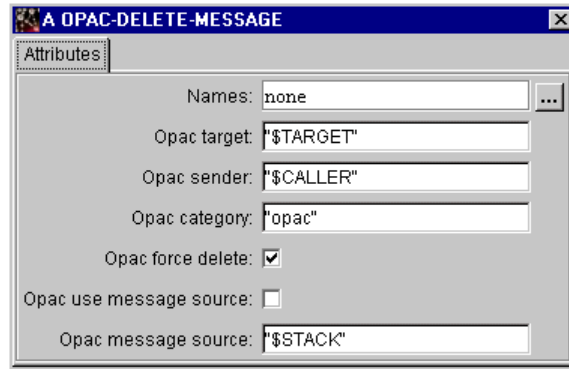
\$stack
\$caller

Force Delete: Use Message Source:

OK Cancel

Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-target	text	"\$TARGET"	Any subclass of opfo-managed-object or opfo-containment-object.
opac-sender	text	"\$CALLER"	Any object.
opac-category	text	"opac"	Any user-defined category. If the value contains (\$) references, the will be resolved.
opac-message-source	text	"\$STACK"	Any valid reference to a message object.
opac-force-delete	symbol	TRUE	TRUE or FALSE.
opac-use-message-source	symbol	FALSE	TRUE or FALSE.

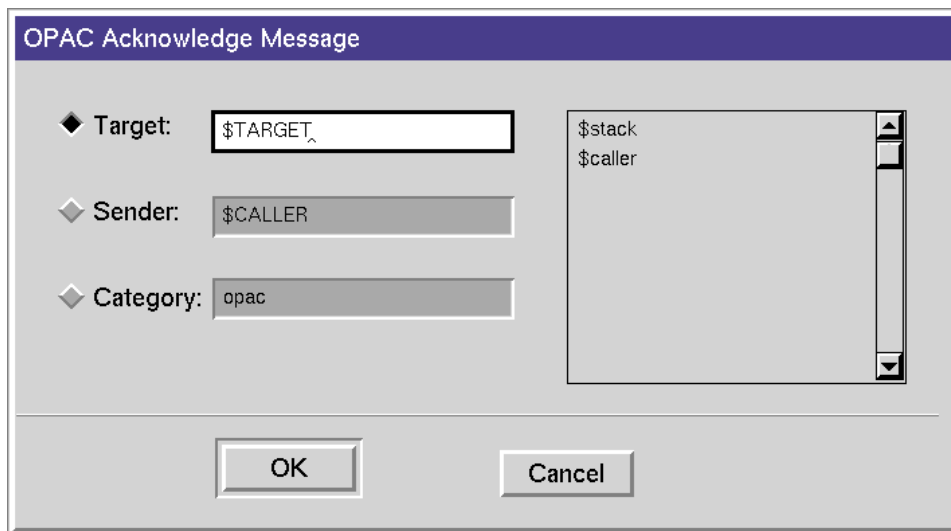
Acknowledge Message



The Acknowledge Message block acknowledges messages based either on specified target, sender, and category; or on a local item variable.

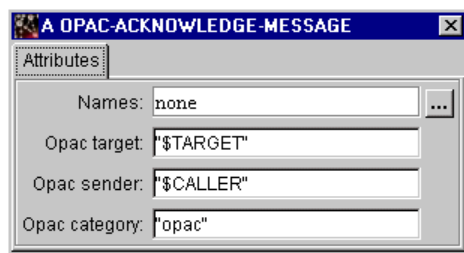
When referencing the message by Target, Sender, and Category, wild cards may be used for the Category.

The local variable can be either a single message object, or a list containing message objects. The local variable can be either a local-item variable connected to the Start Block, or it may be on the token stack.



Properties Dialog

The Properties dialog for the block is shown below:



Attributes

Attribute	Data Type	Default Value	Possible Values
opac-target	text	“\$TARGET”	Any subclass of opfo-managed-object or opfo-containment-object.
opac-sender	text	“\$CALLER”	Any object.
opac-category	text	“opac”	Any user-defined category. If the values contains (\$) references, they will be resolved.

Utilities

Chapter 13: OpEx Dispatch Engine Reference (ODIE)

Provides an overview of the OpEx Dispatch Engine (ODiE) and describes its API.

Chapter 14: Message Parsing Engine (MPE)

Provides reference information on the Message Parsing Engine (MPE).

OpEx Dispatch Engine Reference (ODiE)

Provides an overview of the OpEx Dispatch Engine (ODiE) and describes its API.

Introduction	208
Events	209
Publish Subscribe Mechanism	209
Managers	210
Subscribers	210
Old Event Processing	210
Filters	211
Responses	214
OPAC Blocks for ODiE Events	218
Subscriber Toolbox	221
Classes	223
odie-event	224
odie-event-proxy	226
odie-g2-manager	228
Application Programmer's Interface	230
odie-g2-manager::odie-datastore-add-event-passport-stamp	232
odie-g2-manager::odie-datastore-create-event	233
odie-g2-manager::odie-datastore-delete-event	235
odie-g2-manager::odie-datastore-delete-events	236
odie-g2-manager::odie-datastore-duration-count-query	238
odie-g2-manager::odie-datastore-duration-proxy-query	240
odie-g2-manager::odie-datastore-get-event-attribute-value	242
odie-g2-manager::odie-datastore-get-passport-stamps	243
odie-g2-manager::odie-datastore-set-event-attribute-value	244
odie-g2-manager::odie-datastore-start-time-count-query	245
odie-g2-manager::odie-datastore-start-time-proxy-query	247

odie-manager::odie-manager-add-event-passport-stamp	249
odie-manager::odie-manager-create-event-class	250
odie-manager::odie-manager-delete-event	251
odie-manager::odie-manager-delete-events	252
odie-manager::odie-manager-duration-count-query	254
odie-manager::odie-manager-duration-proxy-query	256
odie-manager::odie-manager-get-event-attribute-value	258
odie-manager::odie-manager-get-passport-stamps	259
odie-manager::odie-manager-passport-meets-include-exclude-criteria	260
odie-manager::odie-manager-post-inform-statement	261
odie-manager::odie-manager-publish-existing-event	262
odie-manager::odie-manager-publish-new-event	263
odie-manager::odie-manager-publish-new-event	264
odie-manager::odie-manager-set-event-attribute	266
odie-manager::odie-manager-start-time-count-query	267
odie-manager::odie-manager-start-time-proxy-query	269
odie-manager::odie-manager-subscribe-event-class	271
odie-manager::odie-manager-substitute-attribute-values	272
odie-manager::odie-manager-unsubscribe	273
odie-manager::odie-manager-unsubscribe-event-class	274



Introduction

OpEx Dispatch Engine (ODiE) is a tool for handling events and responses to events.

Events may be generated internally or received from an external system. ODiE can dispatch events to local or remote subscribers. The history of all events, and the responses to these events, provides an audit trail of how the system attained its state.

An event is any occurrence within the system. The occurrence is typically related to some domain object in the system: this object is the target of the event. The object which creates the event is the sender.

This chapter provides a description of how to use ODiE, as well as a reference, including ODiE:

- Classes
- Application Programmer's Interface
- User Menu Choices

- Relations
- Functions
- Miscellaneous Items

Events

In ODiE, an event is an object of the `odie-event` class. Users can create subclasses of `odie-event` which can be used by subscribers to receive only events that interest them.

To access the attributes of an event, use `odie-manager-get-event-attribute-value`; to modify the attributes of an event use `odie-manager-set-event-attribute-value`. This method will change the value of an existing attribute or, if the attribute does not exist, add a new attribute-value pair to the `additional-data` of the event.

An event can have a passport added by a response using `odie-manager-add-event-passport-stamp`. A passport stamp provides an indication that some action has already taken place on the event. See [Responses](#).

Publish Subscribe Mechanism

ODiE distributes events through a publish/subscribe mechanism. An ODiE manager receives event publication requests and notifies every subscriber of that event class. A subscription may be as general or specific as desired: a subscriber will receive notification of all event classes it specifically subscribes to as well as all subclasses of the specified events.

Any object can publish an event by calling `odie-manager-publish-new-event` (see API for descriptions of the two methods). The API creates an event proxy which is sent to the subscribers; it is also returned to the publisher (the sender of the event). Your application is responsible for consuming the proxy. There is a G2 relation between the proxy(s) and the event (see Event Relations)

Subscribers are notified in an order determined by:

- The module that contains the subscriber (subscribers in lower level modules are notified prior to subscribers in higher level modules based on the G2 established module hierarchy)
- Position in class hierarchy (specific subscribers are notified prior to generic subscribers, i.e., subscribers to event subclasses are notified before subscribers to parent classes)

Managers

Every application must instantiate an `odie-g2-manager`. The manager receives event publication requests and notifies subscribers. The manager also maintains the history of events and provides facilities to query event history.

The manager provides APIs to:

- Create and publish new events
- Query event history
- Modify event attributes

The manager handles deletion of old events via a rule that fires every minute. Events are deleted if they have existed for more than `odie-manager-maximum-event-age`.

Subscribers

An ODiE subscriber specifies interest in one or more classes of events. Each subscriber should have one or more threads of filters and responses attached to it. The responses instruct ODiE to perform some task when an event is received. A filter evaluates the event for specified criteria and passes the event to a response (or another filter) if the criteria are met.

The filters and responses are processed in order:

- 1 Synchronous responses
- 2 Filters
- 3 Asynchronous responses

Subscribers must register with a manager, using `odie-subscriber-initialize`, in order to be notified of events. Subscribers can be "turned off" by calling `odie-subscriber-unsubscribe`.

Old Event Processing

The manager can be configured to automatically delete old events by placing a value in `odie-manager-maximum-event-age`. There is a rule that fires every minute that checks the `minimum-event-age` and deletes all messages that have existed for longer than that value. If `odie-manager-maximum-event-age` has no value (the value = `""`), the rule does not fire.

Filters

A response filter is the mechanism to provide different responses based on event criteria. Response Filters are a class of objects that connect directly to a subscriber, other filters, or responses. Here are some examples of how to use the filters.

Target Class Filter

A target class filter performs the responses attached to it if the target is of the specified class. For example, you could post a lower message if a notebook were a reachability root cause than if a desktop were a root cause. Use `odie-event-class-filter`.

Target Attribute Filter

A target attribute filter performs the responses attached to it if the specified attribute equals the specified value. For example, you could differentiate responses by using `opfo-external-name`. You could post a higher message if the mail server were unreachable than I would if the DC Color printer was unreachable.

Delay Filter

A delay filter waits for the provided amount of time before informing the attached responses. A delay filter may be the basis for alarm escalation. For example, an event is received and a message is posted to the operator. The delay filter then waits for 1 hour. A connected response may check to see that the original message is still there and escalate the event.

Time Filter

A time filter performs the connected responses only if the current time is within the specified limits. For example, send a page to the administrator only if it's after hours.

Query Filter

A query filter performs the responses attached to it if there are a certain number events or messages that have already occurred. For example, query for identical messages prior to publishing a message. If a message already exists, re-published the message with a new time stamp. Another example queries for several identical events (link down) within 10 minutes, and publishes an escalated warning message if this occurs, to indicate a connector problem.

There are eight pre-defined ODiE filters. Some of the filters are logically reversible, which means that every event outside the filter criteria will pass.

Attribute Filter

This filter determines if the event should propagate to the connected odie blocks based on the attribute of the event. If the value specified in the filter is a text, the returned value is converted to text and a regular expression comparison is made. For a description of G2 regular expressions, see "Regular Expression Syntax" in the *G2 Reference Manual*. Use of the standard wildcard characters "*" is an invalid regular expression. An example of a regular expression is: 'abc' to match the sequence of characters abc in the text. The '.' indicates any single character. '(a-z)' indicates any character between 'a' and 'z', with the brackets indicating use the '-' as a meta-character.

If the value specified in the filter is not a text, then the returned value is directly compared to the specified value. If the value is a quantity, an exact match will pass the filter. This filter is reversible by setting the reverse logic attribute to true.

Hour of the Day Filter

Use this filter to pass events created during a certain time interval. The odie-hour-of-the-day-filter passes the event if the current hour is between the start hour and end hour. Use 0-24 to specify hours. Use the reverse-logic attribute set to true to pass events outside the time interval.

Day of the Week Filter

An odie-day-of-the-week-filter checks the current day of the week versus a list of acceptable days. This filter logic is reversible by using a value of true in the reverse logic attribute. The days-of-the-week attribute accepts a sequence containing the weekday names.

Message Historical Query Filter

This query should not be used, but use smh-message-query filter. It is documented here for completeness. To review events no longer existing, use the event query.

This filter passes the event if there are X number of messages in history meeting the filter criteria. The Target, Sender and Category attributes can refer to attributes of the current event using the "\$" notation. The text "any-sender" can be used to match all targets without regard to sender. The message category text can contain the wildcards * to match any number of characters and ? which matches exactly one character. The attribute category-start-position is used with the attribute message-category text string. The category attribute accepts a string to match and the category-start-position defines the first character where the match attempt will start. The operation type attribute accepts greater-than and less-than as arguments. The time-interval takes a string with abbreviated units i.e. 1h = 1 hour, 1s = 1 second. The interval starts with the current time and searches history

for the length of the time interval. See the section on time intervals for more information. The passport stamp takes a text string containing comma separated passport values.

Message Query Filter

This filter passes the event if there are count-threshold number of messages displayed which meet the filter criteria. The Target, Sender and Category attributes can refer to attributes of the current event using the "\$" notation. The text "any-sender" can be used to match all targets without regard to sender. The message category text can contain the wildcards * to match any number of characters and ? which matches exactly one character. The attribute category-start-position is used with the attribute message-category text string. The category attribute accepts a string to match and the category-start-position defines the first character where the match attempt will start. The operation type attribute accepts greater-than and less-than as arguments.

Event Count by Start Time

This filter queries event history for the count of matching events. If the count is greater than or equal to the count-threshold, then the filter passes. The Target, Sender and Class attributes can refer to attributes of the current event using the "\$" notation. The time interval takes a string with abbreviated units i.e. 1h = 1 hour, 1s = 1 second. See the section on time intervals for more information. The interval starts with the current time and searches history for the length of the time interval. The passport stamps to include or exclude must contain comma separated text values. A comparison type of inside will pass all events matching the filter criteria, a comparison type of outside will pass all events not matching the filter criteria.

Passport Filter

Use this filter to pass events that have obtained a certain passport. The odie-passport-filter conditions are met if the event includes all of the stamps listed in the include attribute and none of the stamps in the exclude attribute. The include passport attributes takes a comma separated list of stamps that must be in the event's passport. Entries between commas will be interpreted as a text stamp. Specifying 'ODiE Wildcard' means the event must have at least one passport stamp.

Specifying 'ODiE Wildcard' in the exclude passport stamps means the event can have no passport stamps.

Event Class Filter

Use this filter to pass events of a certain class. Use a text string of comma separated event classes. The strings are converted to symbols to match classes. An event that is named or a subclass of a named event is NOT passed.

Making Your Own Filter Block

All filters are a subclass of `odie-filter`, inherit from this or one of the subclasses. If you want the filter to have the capability to pass all events within the expression or all events outside the expression, or to be logically reversible, then your filter should also inherit from the class `odie-logically-reversible-filter`.

Implement the method `odie-filter-process-connected-blocks@?` with this signature:

```
odie-filter-process-connected-blocks@? (Filter: class odie-filter {an odie-FILTER}, Proxy: class odie-event-proxy {A proxy for an odie-EVENT}, Manager: class ODiE-Manager {an ODiE-Manager}, Client: class ui-client-item {the user client initiating this process}) = (truth-value {true if the evaluation was successful. false otherwise}, text {a description of the error if the operation failed}, truth-value {true if blocks connected at the filters output should be invoked})
```

As an example, to make an `odie-event-class` filter that passes the named events, rather than filtering them, make a new class of filter that inherits from `odie-event-class-filter` and `odie-logically-reversible-filter`. Create a method that calls the superior method. The return value of the superior method should be toggled if the attribute `odie-filter-reverse-logic` is true.

Responses

The response objects are the class of objects that take action. Responses connect directly to subscribers, filters, or other responses. The Delete response is synchronous (it occurs immediately). The other responses are asynchronous and occur in parallel.

Responses can reference attributes of the events that they process using "\$" notation.

Delete Event

A block to delete or clear the current event. Deletes the proxy event given the unique ID of the event. This response procedure is passed a proxy, and deletes the corresponding event. If `$unique-id` is used, then the event being processed is deleted.

Delete Events by Start Time

Deletes all events that match any of the optionally specified event-class, target, sender, passports included, passports excluded, time interval from current time.

Class, target and sender can all reference the current event indirectly. The time interval takes a string with abbreviated units: 1h = 1 hour, 1s = 1 second, 1m = 1 minute and 1s = 1 second. The passport stamp takes a text string containing comma separated passport values. The response-time-comparison attribute has a symbolic value of inside or outside to delete events matching the criteria, or events outside the specified criteria.

This details of this response are to create a list of all proxies matching the criteria defined in the attributes, and then to delete those proxies and their associated events. Each time the passport stamp attributes are edited, the list is parsed into a sequence for internal processing.

G2 Procedure Response

This block invokes the specified G2 procedure. The response creates a proxy and passes it to the procedure, so the user is responsible for deleting this proxy at the conclusion of processing. The attribute odie-response-additional-data is reserved for future use.

The signature of the procedure is:

my-procedure-name

(*response*: class odie-g2-procedure-response, *proxy*: class odie-event-proxy,
manager: class odie-manager, *client*: class ui-client-item)
 -> result: truth-value, error-text: text

where:

- *response* is an odie-g2-procedure-response.
- *proxy* is a proxy for an odie-event.
- *manager* is anan odie-manager.
- *client* is the user client initiating this process.

Create Message

This block creates a smh-message. The target, sender, category, message-text and additional-text can reference the current events attributes with the "\$" notation described in Indirect Reference to Events. Using these references your application can provide a default message containing the event text and class. The message priority must be an integer or text that can be converted to an integer. Message lifetime is the number of seconds before the message is deleted. For no auto-deletion specify "-1". A limited summary of options values is use "-r" to replace

'duplicate' messages, "-a" to append message-text, and "-i" to count the number of messages received. The attribute odie-response-additional-data is reserved for future use.

Clears for or Delete Messages

This block deletes smh-messages. The user should configure which category or categories of message to delete. It should only delete messages of the specified category about the target.

Delete Message

This block deletes messages in the specified category. All smh-messages in the specified server about the target, from the sender and matching the category given. The target, sender and category can reference attributes of the current event using indirect references. The attribute odie-response-additional-data is reserved for future use. The sender and category attributes can be regular expressions. For a description of G2 regular expressions, see "Regular Expression Syntax" in the G2 Reference Manual. Use of the standard wildcard characters "*" is an invalid regular expression. An example of a regular expression is: 'abc' to match the sequence of characters abc in the text. The '.' indicates any single character. '(a-z)' indicates any character between 'a' and 'z', with the brackets indicating use the '-' as a meta-character.

Acknowledge Message

Acknowledges all messages of the given category against the provided target from the provided sender. The target, sender, category and acknowledger are indirect references. The sender and category can be regular expressions. The attribute odie-response-additional-data is reserved for future use. The sender and category attributes can be regular expressions. For a description of G2 regular expressions, see the *G2 Reference Manual*.

Beep

Causes the client to beep the number of times that is entered in the number-of-beeps attribute - as a text string. The number of beeps can be an indirect reference to an attribute of the current event.

The attribute odie-response-additional-data is reserved for future use.

Log_Event

Writes the event to a log file. The log file attribute should contain the full path and the name of the file and can be an indirect reference to an attribute of the current event. The attribute odie-response-additional-data is reserved for future use.

Starting an OPAC Procedure

This block starts the specified OPAC Procedure. If the Subtask Start block requires arguments, the requested arguments are searched for in this order:

- 1 If there is a local parameter defined with the name `odie-manager-name`, then the manager of the current event is used to fill the local parameter.
- 2 If there is a local parameter defined with the name `odie-event-class`, then the event class of the current event is used to fill the local parameter.
- 3 Any event attributes referenced using `$` notation are substituted.
- 4 Any event attributes in the Additional Data attribute referenced using `$` notation are substituted.
- 5 The OPAC start blocks default values are substituted.

Use the following API to start the OPAC procedure:

```
opac-start-task
  (block: class opac-syntax-element, caller: class item,
   target: class object, win: class item, notify: class object,
   arg-list: class item)
  -> token: class opac-token
```

where:

- *block* – The OPAC Start block.
- *caller* – The `sender-id` of the event.
- *target* – The `target-id` of the event.
- *win* – The client UI.
- *notify* – The ODIE block processing the response (`opac-odie-opac-procedure-response`).
- *arg-list* – The argument list of the OPAC start block defined in the `opac-args` attribute.

You can use the following indirect references in the attribute of the any ODIE block:

- `$target` – The target domain object of the event invoking the procedure.
- `$caller` – The sender domain object of the event invoking the procedure.
- `$window` – The window object.
- `$notify` – The OPAC-ODIE start OPAC procedure block.
- `$block` – The block currently being processed.
- `$task` – The task or subroutine currently being run.

- `$stack` – The stack of the OPAC token.

Using Indirect References

The following indirect references in the attribute of a response refer to the attribute of the current event:

- `$event-class` – The class of the event.
- `$unique-id` – The unique-id of the event.
- `$target` – The target that the event was generated against.
- `$sender` – The sender, the generator of the event.
- `$message-text` – The message text of the event.
- `$additional-data` – The additional data of the event.
- `$start-time` – The time the event happened.

The attributes defined in an event's additional-data attribute structure can also be referenced using the "\$" notation. For example, an ODIE blocks message-text attribute can reference the default severity of the event using `$default-severity`.

OPAC Blocks for ODIE Events

OPAC blocks for ODIE events allow are graphical representations of ODIE events. You configure and use these blocks similarly to other OPAC blocks.

Publish New Event

Publishes a new event with the provided information.

Odie Manager Name, Target, Sender, Event-Class, Message Text, and Additional Text can use indirect references to the variables of the OPAC procedure. The attribute values in the structure of Event additional Data can use indirect references. The Event Passport Stamps is a text string of comma separated textual passport stamps.

Publish Event

Re-publish an event given the unique ID of the event.

The indirect references shown are the defaults for this block and assume that the OPAC procedure has been configured with the local parameters "odie-manager-name" and "_odie-event-unique-id" and the opac-args attribute of the start block includes these parameters. Indirect references are discussed in References to

OPAC in ODIE blocks using \$ on page 27 at the end of this section and in Start an OPAC procedure on page 21

Delete Event

Deletes the event given the unique ID of the event.

The indirect references shown are the defaults for this block and assume that the OPAC procedure has been configured with the local parameters "odie-manager-name" and "_odie-event-unique-id" and the opac-args attribute of the start block includes these parameters. Indirect references are discussed in References to OPAC in ODIE blocks using \$ on page 27 at the end of this section and in Start an OPAC procedure on page 21.

Delete Events

Deletes events matching the provided criteria.

The indirect references shown are the defaults for this block. This assumes the OPAC procedure has been configured with the local parameter "odie-manager-name". All attributes except the include & exclude passport-stamps can indirectly reference OPAC token and local parameters as discussed in References to OPAC in ODIE blocks using \$ and in Start an OPAC procedure. The *comparison-type* is either *inside*, describing all matching events inside the time interval, or *outside* describing all matching events outside the time interval.

Get Event Attribute

Retrieves the value of an attribute of the event with the provided unique id and places a parameter containing the value on the OPAC token's stack.

The indirect references shown are the defaults for this block. This assumes the OPAC procedure has been configured with the local parameters "odie-manager-name" and "_odie-event-unique-id" and the opac-args attribute of the start block includes these parameters. Event attribute-name can also indirectly reference OPAC procedure parameters and token attributes. The value of the attribute retrieved is placed in a parameter and put on the top of the OPAC stack.

Values for the references are discussed in References to OPAC in ODIE blocks using \$ on page 27 at the end of this section and in Start an OPAC procedure on page 21.

Set Event Attribute

Sets the value of an attribute of the event with the provided unique id.

The indirect references shown are the defaults for this block. This assumes the OPAC procedure has been configured with the local parameters "odie-manager-

name" and "_odie-event-unique-id" and the opac-args attribute of the start block includes these parameters. The event-attribute-name can also indirectly reference OPAC procedure parameters and token attributes. The attribute-name is given the attribute-value in the matching event.

Values for the references are discussed in References to OPAC in ODIE blocks using \$ at the end of this section and in Start an OPAC procedure on page 21.

Add Passport to Event

Adds a stamp to the event with the provided unique id.

The indirect references shown are the defaults for this block. This assumes the OPAC procedure has been configured with the local parameters "odie-manager-name" and "_odie-event-unique-id" and the opac-args attribute of the start block includes these parameters. The event-passport-stamp can also indirectly reference OPAC procedure parameters and token attributes

Values for the references are discussed in References to OPAC in ODIE blocks using \$ on page 27 at the end of this section and in Start an OPAC procedure on page 21

Count Events

Counts the number of events matching the criteria specified and places the result on the OPAC token stack.

The indirect references shown are the defaults for this block. This assumes the OPAC procedure has been configured with the local parameter "odie-manager-name". All attributes except the include & exclude passport-stamps can indirectly reference OPAC token and local parameters as discussed in References to OPAC in ODIE blocks using \$ and in Start an OPAC procedure. The *comparison-type* is either *inside*, describing all matching events inside the time interval, or *outside* describing all matching events outside the time interval.

Gather Evidence

Places evidence-text of each matching event on the stack in a text parameter. Evidence-text = the event start time + the class of the event + the target of the event + the message text of the event.

The indirect references shown are the defaults for this block. This assumes the OPAC procedure has been configured with the local parameter odie-manager-name. All attributes except the include & exclude passport-stamps can indirectly reference OPAC token and local parameters as discussed in References to OPAC in ODIE blocks using \$ and in Start an OPAC procedure. The *comparison-type* is either *inside*, describing all matching events inside the time interval, or *outside* describing all matching events outside the time interval.

Using Indirect References

Your OPAC ODIE blocks can reference attributes of the token or local OPAC procedure using indirect references. ODIE OPAC substitution variables include the following standard reference passed in as arguments to the API to invoke a graphical procedure including the task or subroutine currently being run:

`$stack` – A stack accessible to the user.

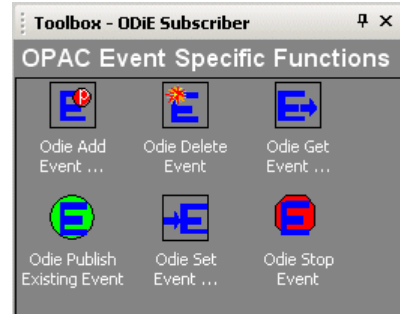
Subscriber Toolbox

ODiE is a graphical tool, which contains palettes of filters and responses that provide the familiar clone, connect, and configure paradigm. To prepare an application to use ODiE, create a new event manager by dragging and dropping from the palette, then configure the manager.

Create a set of subscribers with their associated responses and filters by dragging and dropping the appropriate icons and connecting them., then configure all blocks.

Enable subscribers by calling `odie-subscriber-initialize` for each of them individually or writing an procedure to perform this task for all of your subscribers. You may want to write a startup procedure to handle the subscriptions.

Here are the palettes in the ODIE Subscriber toolbox:



Classes

This section describes the classes defined in the ODiE module:

[odie-event](#)

[odie-event-proxy](#)

[odie-g2-manager](#)

odie-event

Odie-event is the base class of all odie-events.

Class Inheritance Path

odie-event, object, item

Attribute	Description
_odie-event-start-time	When an event is created, its creation time becomes the event start time. this attribute is set by the software, during the event creation process, and should not be modified by the user. <i>Allowable values:</i> Any float
_odie-event-target-id (float)	The target-id should identify the target of the event. The value is passed to odie-manager-publish-new-event. Typically this is the opfo-external-name of some domain object. <i>Allowable values:</i> Any text <i>Default value:</i> ""
_odie-event-sender-id (text)	The sender-id should identify the originator of the event. The value is passed to odie-manager-publish-new-event. Typically this is the opfo-external-name of some object. <i>Allowable values:</i> Any text <i>Default value:</i> ""
_odie-event-message-text	The main contents of any text to be displayed to the user. <i>Allowable values:</i> Any text <i>Default value:</i> ""

Attribute	Description
_odie-event-additional-text	Additional text to be displayed to the user. <i>Allowable values:</i> Any text <i>Default value:</i> ""
_odie-event-additional-data (structure)	A set of Attribute:Value pairs that are used to contain any user-defined attributes for an event. <i>Allowable values:</i> A structure <i>Default value:</i> structure()
_odie-event-passport (sequence)	A set of "stamps" that can be added by any response to this event. Provides an audit-trail and can be used to control responses via the passport stamp filter. <i>Allowable values:</i> A sequence <i>Default value:</i> sequence()

Methods

- To get event attribute values use:
odie-manager::odie-manager-get-event-attribute-value
- To set event attribute values use:
odie-manager::odie-manager-set-attribute

Relations

- _odie-the-event-of relates an event to one or more proxies
- _odie-a-proxy-for relates a proxy to an event

odie-event-proxy

An odie-event-proxy is a smart proxy for an odie-event. As such it has many of the same attributes as the original event. Use odie-manager-get-attribute-value to read any values. Any attributes not visible in the proxy must be retrieved from the original event. To set values of the event, you should use odie-manager-set-attribute-value.

Class Inheritance Path

odie-event-proxy, object, item

An odie-event-proxy(s) is created to represent the original event. The proxy provides a handle to the event.

Attribute	Description
_odie-event-start-time	When an event is created, its creation time becomes the event start time. this attribute is set by the software, during the event creation process, and should not be modified by the user. <i>Allowable values:</i> Any float
_odie-event-target-id	The target-id should identify the target of the event. The value is passed to odie-manager-publish-new-event. Typically this is the opfo-external-name of some domain object. <i>Allowable values:</i> Any text <i>Default value:</i> ""
_odie-event-sender-id (text)	The sender-id should identify the originator of the event. The value is passed to odie-manager-publish-new-event. Typically this is the opfo-external-name of some object. <i>Allowable values:</i> Any text <i>Default value:</i> ""

Attribute	Description
_odie-event-message-text	The main contents of any text to be displayed to the user.
<i>Allowable values:</i>	Any text
<i>Default value:</i>	""
_odie-event-additional-text	Additional text to be displayed to the user.
<i>Allowable values:</i>	Any text
<i>Default value:</i>	""
_odie-event-additional-data (structure)	A set of Attribute:Value pairs that are used to contain any user-defined attributes for an event.
<i>Allowable values:</i>	A structure
<i>Default value:</i>	structure()

Methods

- To get proxy attribute values use:
odie-manager::odie-manager-get-event-attribute-value
- To set proxy/event attribute values use:
odie-manager::odie-manager-set-attribute

odie-g2-manager

An ODIE-G2-Manager is used to dispatch events. A manager also provides facilities to query the events held in its history.

Class Inheritance Path

odie-g2-manager, odie-manager, object, item

Attribute	Description
odie-manager-maximum-event-open-age	
	<i>Allowable values:</i> inherited
	<i>Default value:</i> "1d"
odie-manager-maximum-event-age	Determines the lifetime of events. An odie-g2-manager will delete any event who has existed longer than the given value.
	<i>Allowable values:</i> {"", 1s, 1m, 1h, 1d}
	s = second, m = minute, h = hour, d = day, "" = disable auto-deletion
	Note: Time limits cannot be combined as "1h5m"
	<i>Default value:</i> "1d"

Methods

odie-g2-manager::odie-datastore-start-time-proxy-query

odie-g2-manager::odie-datastore-start-time-count-query

odie-g2-manager::odie-datastore-stop-time-proxy-query

odie-g2-manager::odie-datastore-stop-time-count-query

odie-g2-manager::odie-datastore-duration-proxy-query

odie-g2-manager::odie-datastore-duration-count-query

odie-g2-manager::odie-datastore-create-proxy-for-id

odie-g2-manager::odie-datastore-event-is-a
odie-g2-manager::odie-datastore-stop-event
odie-g2-manager::odie-datastore-get-inheritance-path

User Menu Choices

none

Relations

none

User Menu Choices

odie-turn-on-inform-statements
odie-turn-off-inform-statements

Relations

none

Application Programmer's Interface

This chapter describes the procedures and methods defined in the ODiE module:

[odie-g2-manager::odie-datastore-add-event-passport-stamp](#)

[odie-g2-manager::odie-datastore-create-event](#)

[odie-g2-manager::odie-datastore-delete-event](#)

[odie-g2-manager::odie-datastore-delete-events](#)

[odie-g2-manager::odie-datastore-duration-count-query](#)

[odie-g2-manager::odie-datastore-duration-proxy-query](#)

[odie-g2-manager::odie-datastore-get-event-attribute-value](#)

[odie-g2-manager::odie-datastore-get-passport-stamps](#)

[odie-g2-manager::odie-datastore-set-event-attribute-value](#)

[odie-g2-manager::odie-datastore-start-time-count-query](#)

[odie-g2-manager::odie-datastore-start-time-proxy-query](#)

[odie-manager::odie-manager-add-event-passport-stamp](#)

[odie-manager::odie-manager-create-event-class](#)

[odie-manager::odie-manager-delete-event](#)

[odie-manager::odie-manager-delete-events](#)

[odie-manager::odie-manager-duration-count-query](#)

[odie-manager::odie-manager-duration-proxy-query](#)

[odie-manager::odie-manager-get-event-attribute-value](#)

[odie-manager::odie-manager-get-passport-stamps](#)

[odie-manager::odie-manager-passport-meets-include-exclude-criteria](#)

[odie-manager::odie-manager-post-inform-statement](#)

[odie-manager::odie-manager-publish-existing-event](#)

[odie-manager::odie-manager-publish-new-event](#)

[odie-manager::odie-manager-publish-new-event](#)

[odie-manager::odie-manager-set-event-attribute](#)

[odie-manager::odie-manager-start-time-count-query](#)

[odie-manager::odie-manager-start-time-proxy-query](#)

[odie-manager::odie-manager-subscribe-event-class](#)

[odie-manager::odie-manager-substitute-attribute-values](#)

[odie-manager::odie-manager-unsubscribe](#)

[odie-manager::odie-manager-unsubscribe-event-class](#)

odieg2manager::odiedatastoreaddeventpassportstamp

Synopsis

odieg2manager::odiedatastoreaddeventpassportstamp
(*manager*: class odieg2manager, *proxy*: class odieeventproxy,
eventpassportstamp: text, *client*: ui-client-item)
-> result: truth-value, error: text

Argument	Description
<i>manager</i>	An odieg2manager.
<i>proxy</i>	An odieeventproxy of an odieevent.
<i>eventpassportstamp</i>	The text stamp to add to the passport of the event.
<i>client</i>	The user client initiating this process.

Return Value	Description
<u>result</u>	True if the stamp was added.
<u>error</u>	A description of the error if the operation failed.

Description

Adds the provided text to the passport stamps of the event.

odie-g2-manager::odie-datastore-create-event

Synopsis

odie-g2-manager::odie-datastore-create-event
 (*manager*: class odie-g2-manager, *event-class*: symbol, *target*: text,
sender: text, *message-text*: text, *additional-text*: text, *start-time*: float,
stop-time: float, *passport-stamps*: sequence,
additional-data-structure: structure, *client*: ui-client-item)
 -> result: truth-value, error-text: text, proxy: class odie-event-proxy

Argument	Description
<i>manager</i>	An odie-g2-manager.
<i>event-class</i>	The class of the event to create.
<i>target</i>	The opfo-external-name of the target opfo-domain-object.
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object.
<i>message-text</i>	The main message text used in user communications.
<i>additional-text</i>	Additional information provided in user communications.
<i>start-time</i>	The start time of the event. You should use the current subsecond real time as a default.
<i>stop-time</i>	The stop time of the event. Use -1.0 if the event is not stopped.
<i>passport-stamps</i>	An initial structure of passport stamps.
<i>additional-data-structure</i>	Additional event data as name - value pairs in a structure.
<i>client</i>	The user client initiating this process.

Return Value	Description
<u>result</u>	If a new event is created then true
<u>error-text</u>	A description of the error if the operation failed.
<u>proxy</u>	A proxy for the created odie-event.

Description

Creates an event of with the specified information and stores the event in history.

odieg2-manager::odie-datastore-delete-event

Synopsis

```
odieg2-manager::odie-datastore-delete-event
(manager: class odieg2-manager, proxy: odie-event-proxy,
client: ui-client-item )
-> result: truth-value, error-text: text
```

Argument	Description
<i>manager</i>	An odieg2-manager.
<i>proxy</i>	An odie-proxy of an odie-event.
<i>client</i>	The user client initiating this process.
Return Value	Description
<u>result</u>	True if the attribute was found, false otherwise.
<u>error-text</u>	A description of the error if the operation failed.

Description

Deletes the event.

odie-g2-manager::odie-datastore-delete-events

Synopsis

odie-g2-manager::odie-datastore-delete-events
(*manager*: class odie-g2-manager, *event-class*: symbol, *target*: text,
sender: text, *include-stamps*: sequence, *exclude-stamps*: sequence,
time-attribute: symbol, *start-time*: float, *stop-time*: float,
comparison-type: symbol, *client*: ui-client-item)
-> result: truth-value, error-text: text

Argument	Description
<i>manager</i>	An odie-manager.
<i>event-class</i>	The class of the event to create.
<i>target</i>	The opfo-external-name of the target opfo-domain-object.
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object.
<i>include-stamps</i>	A sequence of stamps to require.
<i>exclude-stamps</i>	A sequence of stamps to prohibit.
<i>time-attribute</i>	Legal values are start-time, stop-time, and duration.
<i>start-time</i>	The beginning of the time interval.
<i>stop-time</i>	The end of the time window to compare.
<i>comparison-type</i>	Valid values are <i>outside</i> or <i>inside</i> , where <i>inside</i> means the start time is between the interval defined by <i>start-time</i> and <i>stop-time</i> .
<i>client</i>	The user client initiating this process.

Return Value	Description
<u>result</u>	True if successful, false otherwise.
<u>error-text</u>	A description of the error if the operation failed.

Description

Deletes the events matching the provided *event-class*, *target*, and *sender*.

odie-g2-manager::odie-datastore-duration-count-query

Synopsis

odie-g2-manager::odie-datastore-duration-count-query
(*manager*: class odie-g2-manager, *event-class*: symbol, *target*: text,
sender: text, *include-stamps*: sequence, *exclude-stamps*: sequence,
smallest-duration: float, *greatest-duration*: float, *comparison-type*: symbol,
client: ui-client-item)
-> result: truth-value, error-text: text, count: integer

Argument	Description
<i>manager</i>	An odie-g2-manager
<i>event-class</i>	The class of the event to query
<i>target</i>	The opfo-external-name of the target opfo-domain-object
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object
<i>includes-tamps</i>	A sequence of stamps to require
<i>exclude-stamps</i>	A sequence of stamps to prohibit
<i>smallest-duration</i>	The smallest duration allowed
<i>greatest-duration</i>	The largest duration allowed
<i>comparison-type</i>	Valid values are outside or inside , where inside means the event's duration is greater than <i>smallest-duration</i> and less than <i>greatest-duration</i>
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	True if the query was successful, false otherwise
<u>error-text</u>	A description of the error if the operation failed
<u>count</u>	The number of matching events

Description

Returns a sequence of events matching the specified criteria.

odie-g2-manager::odie-datastore-duration-proxy-query

Synopsis

odie-g2-manager::odie-datastore-duration-proxy-query
(*manager*: class odie-g2-manager, *event-class*: symbol, *target*: text,
sender: text, *include-stamps*: sequence, *exclude-stamps*: sequence,
smallest-duration: float, *greatest-duration*: float, *comparison-type*: symbol,
proxy-list: class odie-event-proxy-list, *client*: ui-client-item)
-> result: truth-value, error-text: text

Argument	Description
<i>manager</i>	An odie-g2-manager
<i>event-class</i>	The class of the event to query
<i>target</i>	The opfo-external-name of the target opfo-domain-object
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object
<i>include-stamps</i>	A sequence of stamps to require
<i>exclude-stamps</i>	A sequence of stamps to prohibit
<i>smallest-duration</i>	The smallest duration allowed
<i>greatest-duration</i>	The largest duration allowed
<i>comparison-type</i>	Valid values are outside or inside , where inside means the event's duration is greater than <i>smallest-duration</i> and less than <i>greatest-duration</i>
<i>proxy-list</i>	The list to insert matching event proxies at the end of
<i>client</i>	The user client initiating this process

Return Value	Description
<u><i>result</i></u>	True if the query was successful, false otherwise
<u><i>error-text</i></u>	A description of the error if the operation failed

Description

Inserts Proxies at the end of the provided *proxy-list*.

odieg2-manager::odie-datastore-get-event-attribute-value

Synopsis

odieg2-manager::odie-datastore-get-event-attribute-value
(*manager*: class odieg2-manager, *proxy*: odie-event-proxy,
attribute-name: symbol, *client*: ui-client-item)
-> result: truth-value, error-text: text, value: value

Argument	Description
<i>manager</i>	An odieg2-manager
<i>proxy</i>	An odie-proxy of an odie-EVENT
<i>attribute-name</i>	The name of the attribute to retrieve
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	True if the attribute was found, false otherwise
<u>error-text</u>	A description of the error if the operation failed
<u>value</u>	The attributes value or the symbol no-value-found

Description

Returns the value of the event for the given attribute name.

odie-g2-manager::odie-datastore-get-passport-stamps

Synopsis

odie-g2-manager::odie-datastore-get-passport-stamps
 (*manager*: class odie-g2-manager, *proxy*: odie-event-proxy,
client: ui-client-item)
 -> result: truth-value, error-text: text, time-stamps: sequence

Argument	Description
<i>manager</i>	An odie-g2-manager
<i>proxy</i>	A odie-proxy for an odie-EVENT
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	True if the comparison is successful, false otherwise
<u>error-text</u>	A description of the error if the operation failed
<u>time-stamps</u>	A sequence of the stamps in the event's passport

Description

Returns a sequence of stamps in the event's passport. Passport stamps are text values.

odieg2-manager::odie-datastore-set-event-attribute-value

Synopsis

```
odieg2-manager::odie-datastore-set-event-attribute-value  
(manager: class odieg2-manager, proxy: odie-event-proxy,  
attribute-name : symbol, attribute-value: value, client: ui-client-item )  
-> result: truth-value, error-text: text
```

Argument	Description
<i>manager</i>	An odieg2-manager
<i>proxy</i>	An odie-proxy of an odie-event
<i>attribute-name</i>	The name of the attribute to retrieve
<i>attribute-value</i>	The new value for the attribute
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	True if the attribute value is set, false otherwise
<u>error-text</u>	A description of the error if the operation failed

Description

Sets the value of the given attribute in the event.

odie-g2-manager::odie-datastore-start-time-count-query

Synopsis

odie-g2-manager::odie-datastore-start-time-count-query
 (*manager*: class odie-g2-manager, *event-class*: symbol, *target*: text,
sender: text, *include-stamps*: sequence, *exclude-stamps*: sequence,
start-time: float, *stop-time*: float, *comparison-type*: symbol,
client: ui-client-item)
 -> result: truth-value, error-text: text, count: integer

Argument	Description
<i>manager</i>	An odie-g2-manager
<i>event-class</i>	The class of the event to query
<i>target</i>	The opfo-external-name of the target opfo-domain-object
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object
<i>include-stamps</i>	A sequence of stamps to require
<i>exclude-stamps</i>	A sequence of stamps to prohibit
<i>start-time</i>	The beginning of the time interval
<i>stop-time</i>	The end of the time window to compare
<i>comparison-type</i>	Valid values are outside or inside , where inside means the start time is between the interval defined by <i>start-time</i> and <i>stop-time</i> .
<i>client</i>	The user client initiating this process
Return Value	Description
<u>result</u>	True if the query was successful, false otherwise
<u>error-text</u>	A description of the error if the operation failed
<u>count</u>	The number of matching events

Description

Returns a sequence of events matching the specified criteria.

odie-g2-manager::odie-datastore-start-time-proxy-query

Synopsis

```
odie-g2-manager::odie-datastore-start-time-proxy-query
(manager: class odie-g2-manager, event-class: symbol, target: text,
sender: text, include-stamps: sequence, exclude-stamps: sequence,
start-time: float, stop-time: float, comparison-type: symbol,
proxy-list: class odie-event-proxy-list, client: ui-client-item )
-> result: truth-value, error-text: text
```

Argument	Description
<i>manager</i>	An odie-g2-manager
<i>event-class</i>	The class of the event to query
<i>target</i>	The opfo-external-name of the target opfo-domain-object
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object
<i>include-stamps</i>	A sequence of stamps to require
<i>exclude-stamps</i>	A sequence of stamps to prohibit
<i>start-time</i>	The beginning of the time interval
<i>stop-time</i>	The end of the time window to compare
<i>comparison-type</i>	Valid values are outside or inside , where inside means the start time is between the interval defined by <i>start-time</i> and <i>stop-time</i> .
<i>proxy-list</i>	The list to insert matching event proxies at the end of
<i>client</i>	The user client initiating this process

Return Value	Description
<u><i>result</i></u>	True if the query was successful, false otherwise
<u><i>error-text</i></u>	A description of the error if the operation failed

Description

Inserts Proxies at the end of the provided *proxy-list* that meet the given criteria.

odie-manager::odie-manager-add-event-passport-stamp

Synopsis

odie-manager::odie-manager-add-event-passport-stamp
 (*manager*: class odie-manager, *proxy*: class odie-event-proxy,
event-passport-stamp: text, *client*: ui-client-item)
 -> result: truth-value, error-text: text

Argument	Description
<i>manager</i>	An odie-manager
<i>proxy</i>	A proxy of the event to be operated on
<i>event-passport-stamp</i>	The text stamp to add to the passports of the event
<i>client</i>	The user client initiating this process
Return Value	Description
<u>result</u>	The success of the operation
<u>error-text</u>	A description of the error if the operation failed

Description

Adds the provided stamp to the passport of the event.

odie-manager::odie-manager-create-event-class

Synopsis

odie-manager::odie-manager-create-event-class
(*manager*: class odie-manager, *class-name*: symbol, *ancestry*: sequence,
destination-workspace: kb-workspace, *client*: ui-client-item)
-> result: truth-value, error-text: text

Argument	Description
<i>manager</i>	An odie-manager
<i>class-name</i>	The class to create
<i>ancestry</i>	The names (symbols) of the classes making up the inheritance path. The first entry is the Classes direct superior class. The next class is superior class' superior class. The number of ancestors provided is unlimited. If the Ancestry is empty, the direct superior class is assumed to be odie-event. If the last entry in the Ancestry is not odie-event, then the direct superior class of the last ancestor is assumed to be odie-event destination
<i>workspace</i>	The workspace to place newly created classes upon
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	True if the class exists or creating the event class was successful, false otherwise
<u>error-text</u>	A description of the error if the operation failed

Description

Confirms the event named by ClassName exists. If it does not, it creates a class hierarchy using the provided ClassName and Ancestry.

odie-manager::odie-manager-delete-event

Synopsis

odie-manager::odie-manager-delete-event
 (*manager*: class odie-manager, *proxy*: class odie-event-proxy,
client: ui-client-item)
 -> result: truth-value, error-text: text

Argument	Description
<i>manager</i>	An odie-manager
<i>proxy</i>	A proxy of the event to be operated on
<i>client</i>	The user client initiating this process
Return Value	Description
<u>result</u>	The success of the operation
<u>error-text</u>	A description of the error if the operation failed

Description

Deletes the event.

odie-manager::odie-manager-delete-events

Synopsis

odie-manager::odie-manager-delete-events
(*manager*: class odie-manager, *event-class*: symbol, *target*: text,
sender: text, *include-stamps*: sequence, *exclude-stamps*: sequence,
time-attribute: symbol, *start-time*: float, *stop-time*: float,
comparison-type: symbol, *client*: ui-client-item)
-> result: truth-value, error-text: text

Argument	Description
<i>manager</i>	An odie-manager
<i>event-class</i>	The class of the event to create
<i>target</i>	The opfo-external-name of the target opfo-domain-object
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object
<i>include-stamps</i>	A sequence of stamps to require
<i>exclude-stamps</i>	A sequence of stamps to prohibit
<i>time-attribute</i>	Legal values are start-time, stop-time, and duration
<i>start-time</i>	The beginning of the time interval
<i>stop-time</i>	The end of the time window to compare
<i>comparison-type</i>	Valid values are <i>outside</i> or <i>inside</i> , where <i>inside</i> means the start time is between the interval defined by <i>start-time</i> and <i>stop-time</i> .
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	True if successful, false otherwise
<u>error-text</u>	A description of the error if the operation failed

Description

Deletes the events matching the provided *event-class*, *target*, and *sender*.

odie-manager::odie-manager-duration-count-query

Synopsis

odie-manager::odie-manager-duration-count-query
(*manager*: class odie-manager, *event-class*: symbol, *target*: text,
sender: text, *include-stamps*: sequence, *exclude-stamps*: sequence,
smallest-duration float, *greatest-duration*: float, *comparison-type*: symbol,
client: ui-client-item)
-> result: truth-value, error-text: text, count: integer

Argument	Description
<i>manager</i>	An odie-manager
<i>event-class</i>	The class of the event to query
<i>target</i>	The opfo-external-name of the target opfo-domain-object
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object
<i>include-stamps</i>	A sequence of stamps to require
<i>exclude-stamps</i>	A sequence of stamps to prohibit
<i>smallest-duration</i>	The smallest duration allowed
<i>greatest-duration</i>	The largest duration allowed
<i>comparison-type</i>	Valid values are outside or inside , where inside means the event's duration is greater than <i>smallest-duration</i> and less than <i>greatest-duration</i>
<i>client</i>	The user client initiating this process
Return Value	Description
<u>result</u>	True if the query was successful, false otherwise

Return Value	Description
<u><i>error-text</i></u>	A description of the error if the operation failed
<u><i>count</i></u>	The number of events matching the provided criteria

Description

Returns the count of events matching the provided criteria.

odie-manager::odie-manager-duration-proxy-query

Synopsis

odie-manager::odie-manager-duration-proxy-query
(*manager*: class odie-manager, *event-class*: symbol, *target*: text,
sender: text, *include-stamps*: sequence, *exclude-stamps*: sequence,
smallest-duration: float, *greatest-duration*: float, *comparison-type*: symbol,
proxy-list: class odie-event-proxy-list, *client*: ui-client-item)
-> result: truth-value, error-text: text

Argument	Description
<i>manager</i>	An odie-manager
<i>event-class</i>	The class of the event to query
<i>target</i>	The opfo-external-name of the target opfo-domain-object
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object
<i>include-stamps</i>	A sequence of stamps to require
<i>exclude-stamps</i>	A sequence of stamps to prohibit
<i>smallest-duration</i>	The smallest duration allowed
<i>greatest-duration</i>	The largest duration allowed
<i>comparison-type</i>	Valid values are <i>outside</i> or <i>inside</i> , where <i>inside</i> means the event's duration is greater than <i>smallest-duration</i> and less than <i>greatest-duration</i>
<i>proxy-list</i>	The proxy list to append matching proxies to
<i>client</i>	The user client initiating this process
Return Value	Description
<u>result</u>	True if the query was successful, false otherwise
<u>error-text</u>	A description of the error if the operation failed

Description

Appends proxies for the matching events to the end of the provided proxy list.

odie-manager::odie-manager-get-event-attribute-value

Synopsis

```
odie-manager::odie-manager-get-event-attribute-value  
(manager: class odie-manager, proxy: class odie-event-proxy,  
event-attribute-name: symbol, client: ui-client-item )  
-> result: truth-value, error-text: text, value: value
```

Argument	Description
<i>manager</i>	An odie-manager
<i>proxy</i>	A proxy of the event to be operated on
<i>event-attribute-name</i>	The name of the event value to get
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	The success of the operation
<u>error-text</u>	A description of the error if the operation failed
<u>value</u>	The attributes value or the symbol no-value-found

Description

Retrieves a value from an event. Request the name of the attribute you wish to retrieve. `_odie-event-class` retrieves the class of the event.

odie-manager::odie-manager-get-passport-stamps

Synopsis

odie-manager::odie-manager-get-passport-stamps
 (*manager*: class odie-manager, *proxy*: odie-event-proxy,
client: ui-client-item)
 -> result: truth-value, error-text: text, time-stamps: sequence

Argument	Description
<i>manager</i>	An odie-manager
<i>proxy</i>	A proxy of the event to be operated on
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	The success of the operation
<u>error-text</u>	A description of the error if the operation failed
<u>time-stamps</u>	A sequence of the stamps in the event's passport

Description

Returns a sequence of stamps in the event's passport. Passport stamps are text values.

odie-manager::odie-manager-passport-meets-include-exclude-criteria

Synopsis

odie-manager::odie-manager-passport-meets-include-exclude-criteria
(*manager*: class odie-manager, *passport*: sequence,
include-stamps: sequence, *exclude-stamps*: sequence, *client*: ui-client-item)
-> result: truth-value, error-text: text, criteria: truth-value

Argument	Description
<i>manager</i>	An odie-manager
<i>passport</i>	The passport stamps of an event
<i>include-stamps</i>	A sequence of stamps required
<i>exclude-stamps</i>	A sequence of stamps that must not be in the passport
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	True if the evaluation was successful. false otherwise
<u>error-text</u>	A description of the error if the operation failed
<u>criteria</u>	True if passport meets the include/exclude requirements

Description

Determines if the passport stamps meet the include/exclude criteria.

odie-manager::odie-manager-post-inform-statement

Synopsis

odie-manager::odie-manager-post-inform-statement
 (*manager*: class odie-manager, *sender*: item, *statement*: text,
client: ui-client-item)
 -> result: truth-value, error-text: text

Argument	Description
<i>manager</i>	An odie-manager
<i>sender</i>	The entity posting the message
<i>statement</i>	The message to post
<i>client</i>	The user client initiating this process
Return Value	Description
<u>result</u>	True if the post was successful. false otherwise
<u>error-text</u>	A description of the error if the operation failed

Description

Posts a message to the operator.

odie-manager::odie-manager-publish-existing-event

Synopsis

odie-manager::odie-manager-publish-existing-event
(*manager*: class odie-manager, *proxy*: class odie-event-proxy,
client: ui-client-item)
->result: truth-value, error-text: text

Argument	Description
<i>manager</i>	An odie-manager
<i>proxy</i>	A proxy for the event to be re-published
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	The success of the operation
<u>error-text</u>	A description of the error if the operation failed

Description

Republishes an existing event. Only events with changed values or passport stamps should be re-published.

odie-manager::odie-manager-publish-new-event

Synopsis

odie-manager::odie-manager-publish-new-event

(*manager*: class odie-manager, *event-class*: symbol, *target*: text, *sender*: text, *message-text*: text, *additional-text*: text, *client*: ui-client-item)

-> result: truth-value, error-text: text, proxy: class odie-event-proxy

Argument	Description
<i>manager</i>	An odie-manager
<i>event-class</i>	The class of the event to create
<i>target</i>	The opfo-external-name of the target opfo-domain-object
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object
<i>message-text</i>	The main message text used in user communications
<i>additional-text</i>	Additional information provided in user communications
<i>client</i>	The user client initiating this process
Return Value	Description
<u>result</u>	The success of the requested operation
<u>error-text</u>	A description of the error if the operation failed
<u>proxy</u>	A proxy for the new event

Description

Creates and publishes a new event. This API creates an event with no additional data associated with the event.

odie-manager::odie-manager-publish-new-event

Synopsis

odie-manager::odie-manager-publish-new-event
(*manager*: class odie-manager, *event-class*: symbol, *target*: text, *sender*: text, *message-text*: text, *additional-text*: text, *start-time*: float, *stop-time*: float, *passport-stamps*:: sequence, *additional-data-structure* : structure, *client*: ui-client-item)
-> result: truth-value, error-text: text, proxy: class odie-event-proxy

Argument	Description
<i>manager</i>	An odie-manager
<i>event-class</i>	The class of the event to create
<i>target</i>	The opfo-external-name of the target opfo-domain-object
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object
<i>message-text</i>	The main message text used in user communications
<i>additional-text</i>	Additional information provided in user communications
<i>start-time</i>	The start time of the event. You should use the current subsecond real time as a default
<i>stop-time</i>	The stop time of the event. Use -1.0 if the event is not stopped
<i>passport-stamps</i>	An initial structure of passport stamps.
<i>additional-data-structure</i>	Additional event data as name - value pairs in a structure
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	The success of the requested operation
<u>error-text</u>	A description of the error if the operation failed
<u>proxy</u>	A proxy for the new event

Description

Creates and publishes a new event. If the stop time > the start time and ODIE Event Stopped is not in the passport, then ODIE Event Stopped will be added to the passport. If ODIE Event Stopped is in the passport and the stop time < the start time, then the stop time will be set to the start time.

odie-manager::odie-manager-set-event-attribute

Synopsis

odie-manager::odie-manager-set-event-attribute
(*manager*: class odie-manager, *proxy*: class odie-event-proxy,
event-attribute-name: symbol, *event-attribute-value*: value,
client: ui-client-item)
-> result: truth-value, error-text: text, text

Argument	Description
<i>manager</i>	An odie-manager
<i>proxy</i>	A proxy of the event to be operated on
<i>event-attribute-name</i>	The name of the event value to get
<i>event-attribute-value</i>	The new value for the event attribute
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	The success of the operation
<u>error-text</u>	A description of the error if the operation failed

Description

Sets a value of the event.

odie-manager::odie-manager-start-time-count-query

Synopsis

odie-manager::odie-manager-start-time-count-query
 (*manager*: class odie-manager, *event-class*: symbol, *target*: text,
sender: text, *include-stamps*: sequence, *exclude-stamps*: sequence,
start-time: float, *stop-time*: float, *comparison-type*: symbol,
client: ui-client-item)
 -> result: truth-value, error-text: text, count: integer

Argument	Description
<i>manager</i>	An odie-manager
<i>event-class</i>	The class of the event to query
<i>target</i>	The opfo-external-name of the target opfo-domain-object
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object
<i>include-stamps</i>	A sequence of stamps to require
<i>exclude-stamps</i>	A sequence of stamps to prohibit
<i>start-time</i>	The beginning of the time interval
<i>stop-time</i>	The end of the time window to compare
<i>comparison-type</i>	Valid values are Outside or Inside. Inside means the start time is between the interval defined by <i>start-time</i> and <i>stop-time</i> .
<i>client</i>	The user client initiating this process
Return Value	Description
<u>result</u>	True if the query was successful, false otherwise

Return Value	Description
<u><i>error-text</i></u>	A description of the error if the operation failed
<u><i>count</i></u>	The number of events matching the provided criteria

Description

Returns the count of events matching the provided criteria.

odie-manager::odie-manager-start-time-proxy-query

Synopsis

odie-manager::odie-manager-start-time-proxy-query
 (*manager*: class odie-manager, *event-class*: symbol, *target*: text, *sender*: text,
include-stamps: sequence, *exclude-stamps*: sequence, *start-time*: float,
stop-time: float, *comparison-type*: symbol,
proxy-list: class odie-event-proxy-list, *client*: ui-client-item)
 -> result: truth-value, error-text: text

Argument	Description
<i>manager</i>	An odie-manager
<i>event-class</i>	The class of the event to query
<i>target</i>	The opfo-external-name of the target opfo-domain-object
<i>sender</i>	The opfo-external-name of the sender opfo-domain-object
<i>include-stamps</i>	A sequence of stamps to require
<i>exclude-stamps</i>	A sequence of stamps to prohibit
<i>start-time</i>	The beginning of the time interval
<i>stop-time</i>	The end of the time window to compare
<i>comparison-type</i>	Valid values are Outside or Inside. Inside means the start time is between the interval defined by <i>start-time</i> and <i>stop-time</i> .
<i>proxy-list</i>	The proxy list to append matching proxies to
<i>client</i>	The user client initiating this process
Return Value	Description
<u>result</u>	True if the query was successful, false otherwise
<u>error-text</u>	A description of the error if the operation failed

Description

Appends proxies for the matching events to the end of the provided proxy list.

odie-manager::odie-manager-subscribe-event-class

Synopsis

```
odie-manager::odie-manager-subscribe-event-class
(manager: class odie-manager, subscriber: class odie-subscriber,
event-class: symbol, subscriber-include-stamps: sequence,
subscriber-exclude-stamps: sequence, client: ui-client-item )
-> result: truth-value, error-text: text
```

Argument	Description
<i>manager</i>	An odie-manager
<i>subscriber</i>	An odie-subscriber
<i>event-class</i>	The class of event
<i>subscriber-include-stamps</i>	A sequence of the required stamps
<i>subscriber-exclude-stamps</i>	A sequence of the stamps that must not be present
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	True if subscribing was successful, false otherwise
<u>error-text</u>	A description of the error if the operation failed

Description

Adds a subscription to the provided *event-class* for the Subscriber.

odie-manager::odie-manager-substitute-attribute-values

Synopsis

odie-manager::odie-manager-substitute-attribute-values
(*manager*: class odie-manager, *proxy*: class odie-event-proxy,
original-text: text, *client*: ui-client-item)
-> result: truth-value, error-text: text, new-text: text

Argument	Description
<i>manager</i>	An odie-manager
<i>proxy</i>	A proxy of the event to be operated on
<i>original-text</i>	The original text.
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	True if the graphical blocks were processed Successfully, false otherwise
<u>error-text</u>	A description of the error if the operation failed
<u>new-text</u>	The text with attribute values substituted

Description

Substitutes attribute values.

odie-manager::odie-manager-unsubscribe

Synopsis

```
odie-manager::odie-manager-unsubscribe
(manager: class odie-manager, subscriber: class odie-subscriber,
client: ui-client-item )
-> result: truth-value, error-text: text
```

Argument	Description
<i>manager</i>	An odie-manager
<i>subscriber</i>	An odie-subscriber
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	True if subscribing was successful, false otherwise
<u>error-text</u>	A description of the error if the operation failed

Description

Adds a subscription to the provided *event-class* for the Subscriber.

odie-manager::odie-manager-unsubscribe-event-class

Synopsis

```
odie-manager::odie-manager-unsubscribe-event-class  
(manager: class odie-manager, subscriber: class odie-subscriber,  
event-class: symbol, client: ui-client-item )  
-> result: truth-value, error-text: text
```

Argument	Description
<i>manager</i>	An odie-manager
<i>subscriber</i>	An odie-subscriber
<i>event-class</i>	The class of event
<i>client</i>	The user client initiating this process

Return Value	Description
<u>result</u>	True if subscribing was successful, false otherwise
<u>error-text</u>	A description of the error if the operation failed

Description

Removes a subscription to the provided *event-class* for the Subscriber.

Message Parsing Engine (MPE)

Describes the Message Parsing Engine (MPE).

Introduction **276**

General Information **276**

 The OMPE String Receiver **276**

 Message Filter **278**

Message Parsing Engine Palette Blocks **279**

 Conclude Blocks **279**

 Debug Blocks **284**

 Decision Blocks **284**

 Integrity Subsystem Blocks **285**

 Message Handling **287**

 Terminal Blocks **287**

Classes **288**

 mpe-message-filter **289**

 mpe-pause-block **292**

 mpe-procedure-conclude-block **294**

 mpe-single-match-decision-block **296**

 mpe-single-regex-conclude-block **298**

 mpe-start-end-match-decision-block **300**

 mpe-start-end-of-text-conclude-block **302**

 mpe-start-end-regex-conclude-block **304**

 mpe-start-of-text-to-end-regex-conclude-block **306**

 mpe-static-conclude-block **308**

 mpe-string-position-block **310**

 mpe-string-receiver **312**

 mpe-terminal-block **313**

 mpe-text-buffer **314**

 mpe-word-line-conclude-block **316**

 create-message-block **318**

 ompe-delete-message-block **320**

 ompe-opac-subtask-start-block **322**

 ompe-string-receiver **324**

Application Programmer's Interface	326
mpe-current-real-time-as-time-stamp	326
mpe-text-buffer::mpe-add-text-to-buffer	327
mpe-text-buffer::mpe-clear-buffer	327
User Menu Choices	328
mpe-clear-buffer	328
mpe-show-buffer	328
mpe-turn-debugging-off	328
mpe-turn-debugging-on	329
ompe-go-to-procedure	329
Relations	330
_mpe-from-message-filter	330
_mpe-from-text-buffer	330



Introduction

The Integrity Message Parsing Engine (MPE) is a graphical language specifically for parsing text messages.

General Information

This section provide general information on using the Integrity Message Parsing Engine (MPE).

The MPE allows you to easily design a message parser, using convenient graphical blocks for construction. No looping is allowed in the string of blocks.

Note Error-handling procedures can vary and depend on the application of the MPE. For this reason, the MPE does not include built-in error-handling procedures. The user should build all necessary error-handling procedures that are specific to the application.

The OMPE String Receiver

The string receiver object is an object that is passed through the filter blocks, mpe-filter-block, to hold the identified message, to hold local variables, and to control

block execution. You can think of the attributes of the string receiver as items that are populated or built as the string receiver passes through the parsing routine.

The developer or administrator can determine how these become populated. Furthermore, you can subclass this object and define additional attributes. Ompe-string-receiver attributes include the following:

- **ompe-category** - Reflects the category of the incoming text i.e., critical, etc. allowing for message correlation.
- **ompe-message-text** - Text that reflects the desired message that can be displayed in the browser via the opac message block.
- **ompe-additional-text** - Text that reflects the desired additional text to be displayed in the browser via the opac message block.

```
sequence (structure ( attribute-name: the symbol ompe-sender,
                      public: true,
                      allowable-values: "Any text",
                      description: "The opfo-external-name of the sender
                                opfo-domain-object."),
            structure ( attribute-name: the symbol ompe-target,
                      public: true,
                      allowable-values: "Any text",
                      description: "The opfo-external-name of the target
                                opfo-domain-object."),
            structure ( attribute-name: the symbol ompe-category,
                      public: true,
                      allowable-values: "Any text",
                      description: "the smh-message-category"),
            structure ( attribute-name: the symbol ompe-message-text,
                      public: true,
                      allowable-values: "Any text",
                      description: ""),
            structure ( attribute-name: the symbol ompe-additional-text,
                      public: true,
                      allowable-values: "Any text",
                      description: ""),
            structure ( attribute-name: the symbol mpe-receiver-text,
                      public: true,
                      allowable-values: "inherited",
                      description: "The identified message text."),
            structure ( attribute-name: the symbol mpe-working-text,
                      public: true,
                      allowable-values: "inherited",
                      description: "A text area for any processing requiring more
                                that one block."),
            structure ( attribute-name: the symbol mpe-description,
                      public: true,
```

allowable-values: "inherited",
description: "See _mpe-object"))

Message Filter

The following attributes determine what the filter will accept:

- `mpe-start-expression` - The text string must start with this expression.
- `mpe-contains-expression` - The text string must contain this particular expression.
- `mpe-end-expression` - The end of the string is determined by this expression.

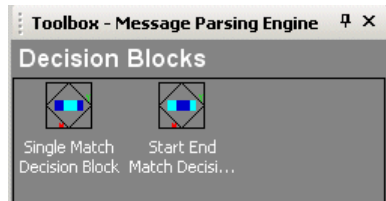
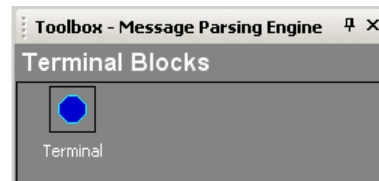
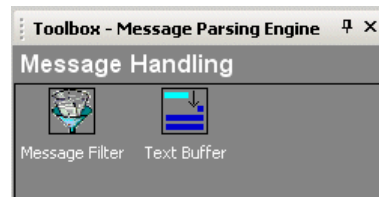
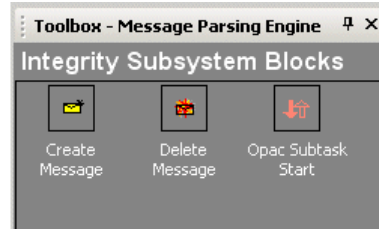
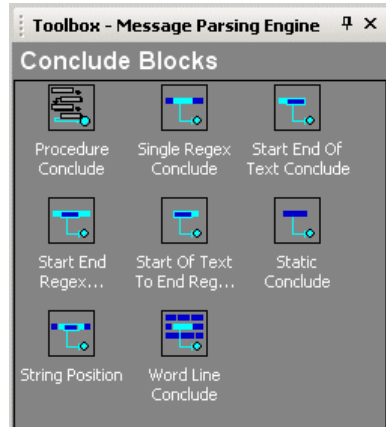
Any of the following elements can be used to build expressions:

- `<alphanumeric>` refers to any character or number
- `<alphabetic>` refers to any character
- `<numeric>` refers to any number
- `+` refers to multiple occurrences of anything that precedes it
- `.` refers to anything (a wild card character)
- `*` refers to 0 or more of the previous thing

One filter can accept input from more than one buffer at a time. Furthermore, you may connect filters in series. The text will travel from one filter to the next until it finds a filter that will accept its format. If the filter or filters do not accept incoming text, then the text will sit in the buffer and will prevent other text from leaving the buffer. You might want to consider connecting a filter at the end of the filter series (if applicable) as a 'catch-all' filter so that the text will leave the buffer.

Message Parsing Engine Palette Blocks

Here are the palettes in the Message Parsing Engine toolbox:



Conclude Blocks

Procedure Conclude



This block launches a procedure that is designed to accept two arguments: an object of procedure block class and an object of the string receiver class. It returns a value that either pre-appends, overwrites or appends any string receiver object attribute that you specify in this block's mpe destination attribute.

- You must include the procedure name in the mpe procedure name attribute.
- Any procedure that is called must accept two arguments: the procedure conclude block object and the string receiver object.
- Any procedure that is called must return a string (even if it is empty).
- The mpe destination attribute determines the return string's destination.
- The mpe write mode determines if the string will append, pre-append or overwrite the text that exists in the mpe destination.

Single Regex Conclude



This block searches from a determined point within a string for an expression that matches a pattern and extracts that expression from the string.

Start End Of Text Conclude



This block extracts a string from the source text based on a beginning expression indicated via the mpe start regex expression. It will extract the part of the text that follows the mpe start regex expression. The extracted string will either include or exclude the mpe start regex expression based on the mpe extract mode (inclusive or exclusive).

Start End Regex Conclude



This block extracts a string from the text based on a beginning expression and ending expression. You may either include or exclude the beginning and ending expression in the extracted string by specifying either "inclusive" or "exclusive" via this block's mpe extract mode attribute.

Start Of Text To End Of Regex Conclude



This block extracts a string from the text based on a beginning search point in the text and an ending expression. You may either include or exclude the ending expression in the extracted string by specifying either "inclusive" or "exclusive" via this block's mpe extract mode attribute. In other words, this block will remove any text that follows the mpe end regex expression, and keep the text up to the mpe end regex expression.

Static Conclude



This block pre-appends, appends, or overwrites a value (string) to any receiver object attribute.

String Position



This block extracts a string from incoming text based on the location of the beginning of the string and the location of the end of the string. For example, if the start position is 1 and the end position is 3, then the string to be extracted will include all characters between and including the first and third character in the text.

String Receiver



This block receives the string.

Attribute	Description
mpe-description	Describes the purpose of this receiver object
mpe-receiver-text	Text that this object receives or currently 'owns'
mpe-working-text	Can be used for further parsing, etc.
ompe-sender	Refers to the opfo-external name of the sender associated with the incoming text. Allowable value is any text.
ompe-sender	The opfo-external-name of the sender opfo-domain-object as a sequence: public:(structure (attribute-name: the symbol ompe-sender, true))
ompe-target	Refers to the opfo-external name of the target associated with the incoming text
ompe-category	Reflects the category of the incoming text i.e. critical, etc. allowing for message correlation
ompe-message-text	Text that reflects the desired message that can be displayed in the browser via the opac message block
ompe-additional-text	Text that reflects the desired additional text to be displayed in the browser via the opac message block.

```

structure ( attribute-name: the symbol ompe-target,
            public: true,
            allowable-values: "Any text",
            description: "The opfo-external-name of the target
                opfo-domain-object."),
structure ( attribute-name: the symbol ompe-category,
            public: true,
            allowable-values: "Any text",
            description: "the smh-message-category"),
structure ( attribute-name: the symbol ompe-message-text,
            public: true,
            allowable-values: "Any text",
            description: ""),
structure ( attribute-name: the symbol ompe-additional-text,
            public: true,
            allowable-values: "Any text",
            description: ""),
structure ( attribute-name: the symbol mpe-receiver-text,
            public: true,
            allowable-values: "inherited",
            description: "The identified message text."),
structure ( attribute-name: the symbol mpe-working-text,
            public: true,
            allowable-values: "inherited",
            description: "A text area for any processing requiring more than
                one block."),
structure ( attribute-name: the symbol mpe-description,
            public: true,
            allowable-values: "inherited",
            description: "See _mpe-object"))

```

Word Line



This block extracts a word from incoming text based on the word's location within the text, that is, the line that it is on and the word position it holds in the line, for example, first word, second word, etc.

Debug Blocks

Pause



This block can be used for debugging. It causes the parsing routine to pause as each string receiver object passes through the pause block. You can control the pause duration via the pause block's delay attribute.

Decision Blocks

Single Match Decision



This block determines if a match has been made by comparing a source string with a user specified string. The attribute, mpe match source should be a mpe string receiver object attribute. The user must also specify the search position within the source in question, for example the first character in the string.

Start End Of Match Decision



This block determines if a match has been made by comparing a source string (mpe-match-source) with a user specified search position (for example, the value "1" indicates the first character in the string), beginning of string value and end of string value. The search will take place by traversing the text from left to right. The source should be an mpe string receiver object attribute.

Integrity Subsystem Blocks

Create Message



This block sends a message to a message server that you specify in the Message Server attribute. It creates a message using attributes of the String Receiver. Attributes are described in the following table:

Attribute	Description
ompe-message-server	The name of the message server to create the message in. The value can be any symbol.
ompe-category	The category of the message created. This value is over written if the ompe-category of the ompe-string-receiver has a value. The value can be any text.
ompe-options	"The standard options for creating an smh-message. e.g. @"-r@",@" -a@",@"-i@",@"-noack@",@"-nohist@"". The value can be any text.
ompe-priority	The priority of the message to create. The value can be any integer.
ompe-lifetime	"The length of time in seconds the created message should exist. (-1 = indefininte.)" The value can be any integer.
mpe-description	For your notation.

Delete Message

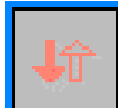


This block deletes messages that fit a category from one, multiple or all server(s) in question. You specify the category in this block's 'ompe category to delete' attribute. You specify the server in this block's 'ompe message server' attribute. It

will determine what messages to delete by matching the target and sender that are specified in the string receiver attributes. Attributes are shown in the following table:

Attribute	Description
ompe-message-server	The name of a message server. Specifying all as the name of the message server deletes messages matching the target, sender, and category to delete from all message servers. The value can be any symbol.
ompe-category-to-delete	The category of message to delete. The value can be any symbol.
mpe-description	For your notation. The value can be any text.

Opac Subtask Start



This block spawns the opac procedure named in the ompe-opac-subtask-start-name attribute of this block. The target and sender are provided by the ompe-string-receiver. Local arguments are also passed to the opac procedure. The local arguments passed are the values of the string receiver attributes listed in the ompe-local-arguments of the ompe-opac-subtask-start. Attributes are shown in the following table:

Attribute	Description:
ompe-opac-subtask-start-name	The name of the opac-subtask-start block. The value can be any symbol.
ompe-local-arguments	"Attribute names of the receiver object" description: "A comma separated list of string receiver attribute names. Each entry will be passed in the specified order to the opac routine as a local argument."
mpe-description	"inherited" description: for your notation

Message Handling

Message Filter



This block accepts text from the text buffer that matches a pattern that you define. It can accept text from more than one text buffer at a time. The message filter has an attribute, `mpe-string-receiver-class`, that requires you to specify the receiver object that will carry the incoming text through the parsing routine.

Text Buffer



The first message handling block necessary to begin a parsing routine.

Give it a name and make certain that the maximum buffer length does not exceed 64,000.

Terminal Blocks

Terminal



This block is the last block necessary to complete the parsing routine.

Classes

This section describes the classes defined in the Message Parsing Engine module:

[mpe-message-filter](#)

[mpe-pause-block](#)

[mpe-procedure-conclude-block](#)

[mpe-single-match-decision-block](#)

[mpe-single-regex-conclude-block](#)

[mpe-start-end-match-decision-block](#)

[mpe-start-end-of-text-conclude-block](#)

[mpe-start-end-regex-conclude-block](#)

[mpe-start-of-text-to-end-regex-conclude-block](#)

[mpe-static-conclude-block](#)

[mpe-string-position-block](#)

[mpe-string-receiver](#)

[mpe-terminal-block](#)

[mpe-text-buffer](#)

[mpe-word-line-conclude-block](#)

[create-message-block](#)

[ompe-delete-message-block](#)

[ompe-opac-subtask-start-block](#)

[ompe-string-receiver](#)

mpe-message-filter

A message filter looks for text expressions in a text buffer (`mpe-text-buffer`). A message is identified by matching start and end regex expressions in the text buffer. Some messages require an additional matching contained expression.

An optional feature allows moving any strings rejected by the filter to a text-list attribute in the filter. To specify this option, use the `MPE-filter-rejects-to-filter-rejects-buffer` truth-value attribute of the `mpe-message-filter` object. When this attribute is set to `true`, if a string is rejected upon a no-match with the `contains-expression`, then the reject string is copied into the `mpe-filter-rejects-buffer`. Parsing will continue. An additional attribute for the `mpe-filter`, `mpe-filter-rejects-buffer-max-entries` (initially set to 100) sets the number of maximum entries to the buffer. The customer can set `max-entries` to any number they wish. (If you are tracking entries manually, over a few hundred may be excessive.)

Class Inheritance Path

`mpe-message-filter`, `_mpe-object`, `object`, `item`

Attributes

Attribute	Description
mpe-start-expression	The regex expression which starts a message. <i>Allowable values:</i> Any text <i>Default value:</i> ""
mpe-contains-expression	A regex expression to be found between the start and end expressions. <i>Allowable values:</i> Any text <i>Default value:</i> "."
mpe-end-expression	The regex expression identifying the end of a message. <i>Allowable values:</i> Any text <i>Default value:</i> ""

Attribute	Description
mpe-string-receiver-class	When a message is identified by this mpe-message-filter, an instance of a string receiver named by this class name is created. <i>Allowable values:</i> The class-name of mpe-string-receiver <i>Default value:</i> MPE-STRING-RECEIVER
mpe-description	See _mpe-object <i>Allowable values:</i> inherited <i>Default value:</i> ""
mpe-filter-rejects-to-filter-rejects-buffer	See _mpe-object <i>Allowable values:</i> truth values <i>Default value:</i> false
mpe-filter-rejects-buffer-max-entries	See _mpe-object <i>Allowable values:</i> integer <i>Default value:</i> 100
mpe-filter-rejects-to-bottom	Used to put a rejected string to the bottom of the filter buffer so that it may be processed again after all of the strings remaining in the buffer are processed first. <i>Allowable values:</i> truth-value <i>Default value:</i> false

Attribute	Description
mpe-filter-rejects-to-filter-rejects-buffer	Used to put a rejected string directly into a rejects buffer set aside for isolating the rejected strings. This will be for the user to parse later in another block or as a record of the strings that are rejected. <i>Allowable values:</i> truth-value <i>Default value:</i> true
mpe-filter-rejects-buffer-max-entries	The allowable number of entries for the filter-rejects-buffer. This can be set by the user. <i>Allowable values:</i> integer <i>Default value:</i> 100
mpe-filter-rejects-buffer	The buffer in the form of a user accessible text-list. <i>Allowable values:</i> text-list <i>Default value:</i> an instance of a mpe-filter-rejects-buffer-list

The mpe-filter-rejects-to-bottom truth-value attribute is true, if a string is rejected upon a no-match with the contains-expression, then the reject string is moved to the bottom of the message-buffer and parsing will continue.

Methods

none

User Menu Choices

configure-message-filter.-.-.

Relations

_mpe-from-message-filter

mpe-pause-block

A pause block simply waits for a period of time before sending control to the next block.

Class Inheritance Path

mpe-pause-block, _mpe-debug-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-delay	The number of seconds to wait before passing control to the next block. <i>Allowable values:</i> Any integer <i>Default value:</i> 5
mpe-debug-on	See mpe-debug-block <i>Allowable values:</i> inherited <i>Default value:</i> true
mpe-description	See _mpe-object <i>Allowable values:</i> inherited <i>Default value:</i> ""

Methods

none

User Menu Choices

configure-pause-block.-.-.

Relations

none

mpe-procedure-conclude-block

This conclude block calls the specified procedure. The return value of the procedure is concluded in the destination. The signature of the procedure is (BLK: class _mpe-filter-block {the filter block currently processing the string receiver}, REC: class mpe-string-receiver {The string receiver}) = (text {the text to return})

Class Inheritance Path

mpe-procedure-conclude-block, _mpe-variable-destination-conclude-block, _mpe-conclude-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-procedure-name	The name of the procedure to call. The procedure will be passed the procedure block and the string receiver. The procedure should return a value. <i>Allowable values:</i> Any symbol <i>Default value:</i> MPE-CURRENT-REAL-TIME-AS-TIME-STAMP
mpe-destination	See mpe-variable-destination-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> MPE-WORKING-TEXT
mpe-write-mode	See mpe-conclude-block <i>Allowable values:</i> PREPEND, APPEND, OVERWRITE <i>Default value:</i> OVERWRITE
mpe-description	See _mpe-object <i>Allowable values:</i> inherited <i>Default value:</i> ""

Methods

none

User Menu Choices

configure-procedure-conclude-block.-.-.

Relations

none

mpe-single-match-decision-block

A decision block based on a single regex expression match.

Class Inheritance Path

mpe-single-match-decision-block, _mpe-decision-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-match-string	The regex expression to find in the match source. <i>Allowable values:</i> Any text <i>Default value:</i> ""
mpe-match-source	See _mpe-decision-block <i>Allowable values:</i> inherited <i>Default value:</i> MPE-RECEIVER-TEXT
mpe-start-search-position	See _mpe-decision-block <i>Allowable values:</i> inherited <i>Default value:</i> 1
mpe-description	See _mpe-object <i>Allowable values:</i> inherited <i>Default value:</i> ""

Methods

none

User Menu Choices

configure-single-match-decision-block.-.-.

Relations

none

mpe-single-regex-conclude-block

A mpe-single-regex-conclude-block extracts text by matching a single regex expression. The matching text is concluded into the destination attribute of the string receiver.

Class Inheritance Path

mpe-single-regex-conclude-block, _mpe-regex-conclude-block, mpe-extracting-conclude-block, _mpe-variable-destination-conclude-block, _mpe-conclude-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-regex-expression	The regex expression to match. <i>Allowable values:</i> Any text <i>Default value:</i> ""
mpe-start-search-position	See mpe-regex-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> 1
mpe-text-source	See mpe-extracting-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> MPE-RECEIVER-TEXT
mpe-destination	See mpe-variable-destination-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> MPE-WORKING-TEXT

Attribute	Description
mpe-write-mode	See mpe-conclude-block
	<i>Allowable values:</i> PREPEND, APPEND, OVERWRITE
	<i>Default value:</i> OVERWRITE
mpe-description	See _mpe-object
	<i>Allowable values:</i> inherited
	<i>Default value:</i> ""

Methods

none

User Menu Choices

configure-single-regex-conclude-block.-.-.

Relations

none

mpe-start-end-match-decision-block

A decision block based on a two regex expression matches.

Class Inheritance Path

mpe-start-end-match-decision-block, _mpe-decision-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-match-start	The first regex expression to find.
<i>Allowable values:</i>	Any text
<i>Default value:</i>	""
mpe-match-end	The second regex expression to find. The expression must occur after the match start.
<i>Allowable values:</i>	Any text
<i>Default value:</i>	""
mpe-match-source	See _mpe-decision-block
<i>Allowable values:</i>	inherited
<i>Default value:</i>	MPE-RECEIVER-TEXT
mpe-start-search-position	See _mpe-decision-block
<i>Allowable values:</i>	inherited
<i>Default value:</i>	1

Attribute	Description
mpe-description	See <code>_mpe-object</code>
	<i>Allowable values:</i> inherited
	<i>Default value:</i> ""

Methods

none

User Menu Choices

configure-start-end-match-decision-block.-.-.

Relations

none

mpe-start-end-of-text-conclude-block

Returns the text starting with the start regex expression to the end of the text source. The start expression is included or excluded based on the extract mode attribute.

Class Inheritance Path

mpe-start-end-of-text-conclude-block, _mpe-extra-text-regex-conclude-block, _mpe-regex-conclude-block, _mpe-extracting-conclude-block, _mpe-variable-destination-conclude-block, _mpe-conclude-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-start-regex-expression	The regular expression defining the start of the text. <i>Allowable values:</i> Any text <i>Default value:</i> ""
mpe-extract-mode	See mpe-extra-text-regex-conclude-block <i>Allowable values:</i> INCLUSIVE, EXCLUSIVE <i>Default value:</i> EXCLUSIVE
mpe-start-search-position	See mpe-regex-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> 1
mpe-text-source	See mpe-extracting-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> MPE-RECEIVER-TEXT

Attribute	Description
mpe-destination	See mpe-variable-destination-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> MPE-WORKING-TEXT
mpe-write-mode	See mpe-conclude-block <i>Allowable values:</i> PREPEND, APPEND, OVERWRITE <i>Default value:</i> OVERWRITE
mpe-description	See _mpe-object <i>Allowable values:</i> inherited <i>Default value:</i> ""

Methods

none

User Menu Choices

configure-start-end-of-text-conclude-block.-.-.

Relations

none

mpe-start-end-regex-conclude-block

This block returns the text found between a starting regex expression and an ending regex expression. The expressions are included or excluded based on the extract mode.

Class Inheritance Path

mpe-start-end-regex-conclude-block, _mpe-extra-text-regex-conclude-block, _mpe-regex-conclude-block, _mpe-extracting-conclude-block, _mpe-variable-destination-conclude-block, _mpe-conclude-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-start-regex-expression	The regular expression defining the start of the text. <i>Allowable values:</i> Any text <i>Default value:</i> ""
mpe-end-regex-expression	The regex expression marking the end of the text to extract. <i>Allowable values:</i> Any text <i>Default value:</i> ""
mpe-extract-mode	See mpe-extra-text-regex-conclude-block <i>Allowable values:</i> INCLUSIVE, EXCLUSIVE <i>Default value:</i> EXCLUSIVE
mpe-start-search-position	See mpe-regex-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> 1

Attribute	Description
mpe-text-source	See mpe-extracting-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> MPE-RECEIVER-TEXT
mpe-destination	See mpe-variable-destination-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> MPE-WORKING-TEXT
mpe-write-mode	See mpe-conclude-block <i>Allowable values:</i> PREPEND, APPEND, OVERWRITE <i>Default value:</i> OVERWRITE
mpe-description	See _mpe-object <i>Allowable values:</i> inherited <i>Default value:</i> ""

Methods

none

User Menu Choices

configure-start-end-regex-conclude-block.-.-.

Relations

none

mpe-start-of-text-to-end-regex-conclude-block

Extracts the text from the beginning of the text buffer to the end expression.

Class Inheritance Path

mpe-start-of-text-to-end-regex-conclude-block,
_mpe-extra-text-regex-conclude-block, _mpe-regex-conclude-block,
_mpe-extracting-conclude-block, _mpe-variable-destination-conclude-block,
_mpe-conclude-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-end-regex-expression	The regular expression defining the end of the text. <i>Allowable values:</i> Any text <i>Default value:</i> ""
mpe-extract-mode	See _mpe-extra-text-regex-conclude-block <i>Allowable values:</i> INCLUSIVE, EXCLUSIVE <i>Default value:</i> EXCLUSIVE
mpe-start-search-position	See _mpe-regex-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> 1
mpe-text-source	See _mpe-extracting-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> MPE-RECEIVER-TEXT

Attribute	Description
mpe-destination	See <code>_mpe-variable-destination-conclude-block</code>
	<i>Allowable values:</i> inherited
	<i>Default value:</i> MPE-WORKING-TEXT
mpe-write-mode	See <code>_mpe-conclude-block</code>
	<i>Allowable values:</i> PREPEND, APPEND, OVERWRITE
	<i>Default value:</i> OVERWRITE
mpe-description	See <code>_mpe-object</code>
	<i>Allowable values:</i> inherited
	<i>Default value:</i> ""

Methods

none

User Menu Choices

configure-start-of-text-to-end-regex-conclude-block.-.-.

Relations

none

mpe-static-conclude-block

Concludes a static value to a single attribute of the mpe-string-receiver.

Class Inheritance Path

mpe-static-conclude-block, _mpe-variable-destination-conclude-block, _mpe-conclude-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-value	The value to conclude
<i>Allowable values:</i>	Any value
<i>Default value:</i>	""
mpe-destination	See mpe-variable-destination-conclude-block
<i>Allowable values:</i>	:inherited
<i>Default value:</i>	MPE-WORKING-TEXT
mpe-write-mode	See mpe-conclude-block
<i>Allowable values:</i>	PREPEND, APPEND, OVERWRITE
<i>Default value:</i>	OVERWRITE
mpe-description	See _mpe-object
<i>Allowable values:</i>	inherited
<i>Default value:</i>	""

Methods

none

User Menu Choices

configure-static-conclude-block.-.-.

Relations

none

mpe-string-position-block

Extracts the text between (inclusive) the start position and the end position in the source text.

Class Inheritance Path

mpe-string-position-block, _mpe-extracting-conclude-block, _mpe-variable-destination-conclude-block, _mpe-conclude-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-start-position	The start position of the text to extract <i>Allowable values:</i> Any integer <i>Default value:</i> 1
mpe-end-position	The end position of the text to extract <i>Allowable values:</i> Any integer <i>Default value:</i> 2
mpe-text-source	See _mpe-extracting-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> MPE-RECEIVER-TEXT
mpe-destination	See _mpe-variable-destination-conclude-block <i>Allowable values:</i> inherited <i>Default value:</i> MPE-WORKING-TEXT
mpe-write-mode	See _mpe-conclude-block

Attribute	Description
<i>Allowable values:</i>	PREPEND, APPEND, OVERWRITE
<i>Default value:</i>	OVERWRITE
mpe-description	See <code>_mpe-object</code>
<i>Allowable values:</i>	inherited
<i>Default value:</i>	""

Methods

none

User Menu Choices

configure-string-position-block.-.-.

Relations

none

mpe-string-receiver

Class Inheritance Path

mpe-string-receiver, _mpe-object, object, item

Attributes

Attribute	Description
-----------	-------------

mpe-receiver-text

Allowable values: Any text

Default value: ""

mpe-working-text

Allowable values: Any text

Default value: ""

mpe-description

Allowable values: Any text

Default value: ""

Methods

none

User Menu Choices

none

Relations

_mpe-from-text-buffer

_mpe-from-message-filter

mpe-terminal-block

Marks the end of processing. The terminal block deletes the token.

Class Inheritance Path

mpe-terminal-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-description	See _mpe-object

Allowable values: inherited

Default value: ""

Methods

none

User Menu Choices

none

Relations

none

mpe-text-buffer

A message text buffer receives text from external sources via the api `mpe-add-text-to-buffer`. Once text is received, the `mpe-text-buffer` tests the message filters (`mpe-message-filter`) connected at an output of the text buffer.

Class Inheritance Path

`mpe-text-buffer`, `_mpe-object`, `object`, `item`

Attributes

Attribute	Description
mpe-data-source-name	A word or words naming the source of the text. (currently not required.)
<i>Allowable values:</i>	Any text
<i>Default value:</i>	""
mpe-data-source-description	A description of the text data source. (currently not required.)
<i>Allowable values:</i>	Any text
<i>Default value:</i>	""
mpe-maximum-buffer-length	The maximum length of text to store in a text buffer. The upper limit of the length of this buffer is the length of a text string in G2 (roughly 64000). If the length of the buffer exceeds this limit, excess characters will be removed from the beginning of the buffer.
<i>Allowable values:</i>	Any integer
<i>Default value:</i>	10000

Attribute	Description
mpe-description	See <code>_mpe-object</code>

Allowable values: inherited

Default value: ""

Methods

`mpe-text-buffer::mpe-add-text-to-buffer`

`mpe-text-buffer::mpe-clear-buffer`

User Menu Choices

`configure-text-buffer.-.-.`

`mpe-clear-buffer`

`mpe-show-buffer`

Relations

`_mpe-from-text-buffer`

mpe-word-line-conclude-block

Extracts a single value represented by a given word on a given line. Words are space delimited and lines are ^J delimited.

Class Inheritance Path

mpe-word-line-conclude-block, _mpe-extracting-conclude-block, _mpe-variable-destination-conclude-block, _mpe-conclude-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
mpe-line-delimiter	
	<i>Allowable values:</i> Any text
	<i>Default value:</i> " "
mpe-line-number	The line the word is on.
	<i>Allowable values:</i> Any integer
	<i>Default value:</i> 1
mpe-word-delimiter	
	<i>Allowable values:</i> Any text
	<i>Default value:</i> " "
mpe-word-number	The word number to extract.
	<i>Allowable values:</i> Any integer
	<i>Default value:</i> 1

Attribute	Description
mpe-text-source	See mpe-extracting-conclude-block
	<i>Allowable values:</i> inherited
	<i>Default value:</i> MPE-RECEIVER-TEXT
mpe-destination	See mpe-variable-destination-conclude-block
	<i>Allowable values:</i> inherited
	<i>Default value:</i> MPE-WORKING-TEXT
mpe-write-mode	See mpe-conclude-block
	<i>Allowable values:</i> PREPEND, APPEND, OVERWRITE
	<i>Default value:</i> OVERWRITE
mpe-description	See _mpe-object
	<i>Allowable values:</i> inherited
	<i>Default value:</i> ""

Methods

none

User Menu Choices

configure-word-line-conclude-block.-.-.

Relations

none

create-message-block

Creates a message using attributes of the ompe-string-receiver and the block.

Class Inheritance Path

ompe-create-message-block, _ompe-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
ompe-message-server	The name of the message server to create the message in. <i>Allowable values:</i> Any symbol <i>Default value:</i> NONE
ompe-category	The category of the message created. This value is overridden if the ompe-category of the ompe-string-receiver has a value. <i>Allowable values:</i> Any text <i>Default value:</i> ""
ompe-options	The standard options for creating an smh-message. e.g. "-r", "-a", "-i", "-noack", "-nohist" <i>Allowable values:</i> Any text <i>Default value:</i> "-r"
ompe-priority	The priority of the message to create. <i>Allowable values:</i> Any integer <i>Default value:</i> 10

Attribute	Description
ompe-lifetime	The length of time in seconds the created message should exist. (-1 = indefinite.)
	<i>Allowable values:</i> Any integer
	<i>Default value:</i> -1
mpe-description	See mpe-object
	<i>Allowable values:</i> inherited
	<i>Default value:</i> ""

Methods

none

User Menu Choices

configure-create-message-block.-.-.

Relations

none

ompe-delete-message-block

Deletes messages from one or all servers matching the target and sender specified in the string receiver and the category to delete in the block.

Class Inheritance Path

ompe-delete-message-block, _ompe-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
ompe-message-server	The name of a message server. Specifying all as the name of the message server deletes messages matching the target, sender, and category to delete from all message servers. <i>Allowable values:</i> Any symbol <i>Default value:</i> ALL
ompe-category-to-delete	The category of message to delete. <i>Allowable values:</i> Any text <i>Default value:</i> ""
mpe-description	See mpe-object <i>Allowable values:</i> inherited <i>Default value:</i> ""

Methods

none

User Menu Choices

configure-delete-message-block.-.-.

Relations

none

ompe-opac-subtask-start-block

Spawns the opac procedure named in the ompe-opac-subtask-start-name of the block. The target and sender are provided by the ompe-string-receiver. Local arguments are also passed to the opac procedure. The local arguments passed are the values of the string receiver attributes listed in the ompe-local-arguments of the ompe-opac-subtask-start-block.

Class Inheritance Path

ompe-opac-subtask-start-block, _ompe-block, _mpe-filter-block, _mpe-object, object, item

Attributes

Attribute	Description
ompe-opac-subtask-start-name	The name of the opac-subtask-start block. <i>Allowable values:</i> Any symbol <i>Default value:</i> NONE
ompe-local-arguments	A comma separated list of string receiver attribute names. Each entry will be passed in the specified order to the opac routine as a local argument. <i>Allowable values:</i> Attribute names of the receiver object <i>Default value:</i> ""
mpe-description	See mpe-object <i>Allowable values:</i> inherited <i>Default value:</i> ""

Methods

none

User Menu Choices

configure-opac-subtask-start-block.-.-.

ompe-go-to-procedure

Relations

none

ompe-string-receiver

An object passed through the filter blocks (mpe-filter-block) to hold the identified message, to hold local variables, and to control block execution.

Class Inheritance Path

ompe-string-receiver, mpe-string-receiver, _mpe-object, object, item

Attributes

Attribute	Description
ompe-sender	The opfo-external-name of the sender opfo-domain-object. <i>Allowable values:</i> Any text <i>Default value:</i> ""
ompe-target	The opfo-external-name of the target opfo-domain-object. <i>Allowable values:</i> Any text <i>Default value:</i> ""
ompe-category	the smh-message-category <i>Allowable values:</i> Any text <i>Default value:</i> ""
ompe-message-text	 <i>Allowable values:</i> Any text <i>Default value:</i> ""

Attribute	Description
ompe-additional-text	
	<i>Allowable values:</i> Any text
	<i>Default value:</i> ""
mpe-receiver-text	The identified message text.
	<i>Allowable values:</i> inherited
	<i>Default value:</i> ""
mpe-working-text	A text area for any processing requiring more than one block.
	<i>Allowable values:</i> inherited
	<i>Default value:</i> ""
mpe-description	See _mpe-object
	<i>Allowable values:</i> inherited
	<i>Default value:</i> ""

Methods

none

User Menu Choices

none

Relations

none

Application Programmer's Interface

This section describes the public procedures and methods defined by Integrity Message Parsing Engine.

[mpe-current-real-time-as-time-stamp](#)

[mpe-text-buffer::mpe-add-text-to-buffer](#)

[mpe-text-buffer::mpe-clear-buffer](#)

mpe-current-real-time-as-time-stamp

Synopsis

```
mpe-current-real-time-as-time-stamp  
  (block: _mpe-filter-block, receiver: mpe-string-receiver )  
  -> timestamp: text
```

Argument	Description
<i>block</i>	The filter block currently processing the string receiver
<i>receiver</i>	The current thread of the block procedure

Return Value	Description
<u><i>timestamp</i></u>	The current real time as a time stamp

Description

Returns the current real time as a time stamp.

mpe-text-buffer::mpe-add-text-to-buffer

Synopsis

```
mpe-text-buffer::mpe-add-text-to-buffer
  (ltb: mpe-text-buffer, text-to-add: text )
```

Argument	Description
<i>ltb</i>	The mpe-text-buffer to add text to
<i>text-to-add</i>	The text to add to the mpe-text-buffer

Description

mpe-add-text-to-buffer is the main api to a mpe-text-buffer. Each call to this api adds the TextToAdd to the buffer and tests the message filters (mpe-message-filter) connected at an output of the text buffer.

mpe-text-buffer::mpe-clear-buffer

Synopsis

```
mpe-text-buffer::mpe-clear-buffer
  (ltb: mpe-text-buffer )
```

Argument	Description
<i>ltb</i>	The mpe-text-buffer to be cleared.

Description

Calling mpe-clear-buffer empties the internal text buffer.

User Menu Choices

This section describes the user menu choices defined by Integrity Message Parsing Engine:

[mpe-clear-buffer](#)

[mpe-show-buffer](#)

[mpe-turn-debugging-off](#)

[mpe-turn-debugging-on](#)

[ompe-go-to-procedure](#)

mpe-clear-buffer

Applicable Class

mpe-text-buffer

Description

mpe-show-buffer

Applicable Class

mpe-text-buffer

Description

mpe-turn-debugging-off

Applicable Class

_mpe-debug-block

Description

Turns off the debugging functionality of the block.

mpe-turn-debugging-on

Applicable Class

_mpe-debug-block

Description

Turns on the debugging functionality of the block.

ompe-go-to-procedure

Applicable Class

ompe-opac-subtask-start-block

Description

Shows the OPAC Subtask Start Block named by the ompe-opac-subtask-start-block at the top center of the screen

Relations

This section describes the relations defined by Integrity Message Parsing Engine.

[_mpe-from-message-filter](#)

[_mpe-from-text-buffer](#)

_mpe-from-message-filter

Properties

Property	Value
inverse	<code>_mpe-the-message-filter-of</code>
first class	<code>mpe-string-receiver</code>
second class	<code>mpe-message-filter</code>
relation type	<code>many-to-one</code>
symmetric?	<code>false</code>
permanent?	<code>false</code>

Description

A relation between a string receiver (`mpe-string-receiver`) and the message filter (`mpe-message-filter`).

_mpe-from-text-buffer

Properties

Property	Value
inverse	<code>_mpe-the-text-buffer-of</code>
first class	<code>mpe-string-receiver</code>
second class	<code>mpe-text-buffer</code>
relation type	<code>many-to-one</code>

Property	Value
symmetric?	false
permanent?	false

Description

A relation between a string receiver (mpe-string-receiver) and the text buffer (mpe-text-buffer). You may wish to reference the text buffer to retrieve the data source name or data source description.

Autodiscovery

Chapter 15: IP Reachability Analyzer (IPRA)

Provides information on the IP Reachability Analyzer (IPRA) module of the Integrity product family.

Chapter 16: Object Reachability Analysis (ORA-TWO)

Provides an overview of the Object Reachability Analyzer (ORA-TWO) product, its setup, and API.

Chapter 17: Domain Export/Import (DXI3)

Provides a description of how to import and update the domain map.

Chapter 18: Open View Map Importer (OVMAP)

Provides an introduction to the Open View Map Importer (OVMAP) module and describes how to install and setup OVMAP .

Chapter 19: Ping Manager

Introduces the Ping Manager and describes how to install, setup, and run the Ping Manager.

IP Reachability Analyzer (IPRA)

Provides information on the IP Reachability Analyzer (IPRA) module of the Integrity product family.

Introduction	335
Setting up G2/IPRA	336
Setting Up the Ping Manager	338
Summary of IPRA Default Behavior	340
Procedures	341



Introduction

The IP Reachability Analyzer (IPRA) product is a starting point for providing reachability analysis, using other products that are part of the Integrity family of products. IPRA provides the foundation upon which the end-user application is based.

The IP Reachability Analyzer provides the configuration items necessary to support the Ping Manager, SNMP trap handling, Object Reachability Analysis (ORA-TWO), and HP OpenView and/or Integrity DXI3-format domain map importation.

Setting up G2/IPRA

To set up G2/IPRA:

- 1 Initializations exist to update predefined parameters automatically.

The initializations are:

- Ipra-synchronous-interface - SNMP get/set/send traps (default = tcp-ip host "localhost" port-number 22044)
- Ipra-event-interface - SNMP receive traps traps (default = tcp-ip host "localhost" port-number 22045)
- Ipra-scheduled-ping-interface- scheduled polling traps (default = tcp-ip host "localhost" port-number 22053)
- Ipra-demand-ping-interface- demand polling traps (default = tcp-ip host "localhost" port-number 22050)
- Ipra-community-ro- community string for performing snmp gets (default = public)
- Ipra-nms-module- module name of network management station (default = ipra)
- Ipra-snmp-timeout- timeout for SNMP gets (default = 20s)
- Ipra-snmp-retry-count- retry count for SNMP gets (default = 10)
- Ipra-use-java-snmp- true if using SNMP Java based bridge, (default = false)
- Ipra-use-management-file- use a .csv file listing managed devices to indicate which snmp devices to monitor (default = " ")
- Ipra-snmp-interface-for-gets- the gsi interface configured for snmp gets/sets (default = ipra-synchronous-interface)
- Ipra-poll-interval- the default poll interval for ipra (default = 600s)
- Ipra-poll-timeout- the timeout for ipra polling (default = 30s)
- Ipra-poll-retries- the number of retries for ipra polling (default = 3)
- Ipra-repoll-interval- the minimum time lapse in seconds between successive polls of an ip card (default = 80s)
- ipra-root-cause-poll-interval- the polling interval for ip cards that are root causes (default = 300)
- ipra-ping-config-file-path and name of configuration file to write (default= \tmp\scheduled - accessed by your-upload-proc, see below)

- 2 "your-upload-proc" sets the management status of interfaces depending on the management state of the containing SNMP device and writes the interface Ping Manager configurations to a file. It also ensures that at least one interface of a managed switch is periodically polled. By default interfaces of a switch are virtual and therefore not connected to anything i.e. unmanaged. This routine concludes that one of these is managed and adds it to the configuration file. It also sets the management status of ports - any port not connected to a managed interface or managed switch (via another port) will be set as unmanaged.
- 3 Each time the `gsi-interface-status` of a `gsi-interface` is reset from 2, a reset procedure initiates a reconnection. If the bridge in question is the `ipra-scheduled-ping-interface` the configuration file is automatically uploaded. In addition it calls a routine to periodically poll all managed ports, only if a procedure invocation for this is not already live.
- 4 Make changes if necessary to the following `ipra` methods (for example, implement logging features, etc.)
 - `Ipra-card::ora-two-poll-node`
 - `Ipra-port::ora-two-poll-node`
 - `Ipra-domain-object::ora-two-node-type`
 - `Ipra-domain-object::ora-two-fail-method`
 - `Ipra-domain-object::ora-two-recover-method`
 - `Ipra-domain-object::ora-two-poll-fail-method`
 - `Ipra-domain-object::ora-two-predicted-fail-method`
 - `Ipra-domain-object::ora-two-poll-root-cause-method`
 - `Ipra-domain-object::ora-two-poll-recover-method`
 - `Ipra-interface::ora-two-collect-terminal-nodes`
 - `Ipra-interface::ora-two-collect-non-terminal-nodes`
 - `Ipra-interface::ora-two-collect-passive-nodes`
 - `Ipra-router::ipra-collect-snmp-device-interfaces`
 - `Ipra-switch-router:: ipra-collect-snmp-device-interfaces`
 - `Ipra-switch:: ipra-collect-snmp-device-interfaces`
 - `Ipra-ora-two-manager-object::ora-two-domain-consistency-check`
 - `Ipra-ora-two-manager-object::ora-two-collect-all-domain-nodes`
 - `Ipra-ora-two-manager-object::ora-two-collect-related-nodes`

Setting Up the Ping Manager

Use the following guidelines for using Ping Manager with IPRA:

- 1 For the ping manager gsi-interface objects in G2, instead of the Ping Manager object PING-MANAGER-INTERFACE, use the IPRA objects IPRA-SCHEDULED-PING-INTERFACE and "IPRA-DEMAND-PING-INTERFACE" provided with IPRA in the Integrity package.
- 2 Setup and remote procedure calls (RPC's) should be used the same as noted in the documentation section on the PING MANAGER.
- 3 Except for the usage of the IPRA GSI-interface objects instead of the Ping Manager gsi-interface object PING-MANAGER-INTERFACE, setup and remote procedure calls (RPC's) should be used the same as noted in the documentation section on the PING MANAGER.
- 4 Start a Ping Manager executable for each GSI-interface that you use in IPRA.
- 5 Ensure that a different port-number is used for each GSI-interface/Ping Manager executable pair. Ensure that no other GSI-interface is using the same port-number. To compare port-numbers, use the G2 Inspect: display a table of every gsi-interface.
- 6 When you start the Ping Manger executable (*pingmgr.exe* for Windows), the version number is displayed.
- 7 The default port number is 2500 if you start pingmgr without a port argument. The command with port number to be entered for IPRA-SCHEDULED-PING-INTERFACE and IPRA-DEMAND-PING-INTERFACE are respectively:

```
pingmgr 22050
```

```
pingmgr 22052
```

- 8 These line commands for Pingmgr executable can use the available ping-manager argument form. This includes the timeout for the ping manager - which is set at the startup of the pingmgr executable.

For Unix, the ping manager command is:

```
$ ./pingmgr {port-number} {-t[imeout] n} {-r[etries] m}
```

For the Windows command prompt, the command is:

```
Pingmgr {port-number} {-v[erbose]} {-t[imeout] n} {-r[etries] m}
```

Troubleshooting an IPRA Ping Manager GSI-Interface

Running the Ping Manager requires a `gsi-interface` status of 2 on the `gsi-interface` objects. Reference the GSI interfaces found on the `ipra kb` workspaces `lpra-Top-level ws > Application Objects > Receiver objects > Ping/SNMP interface`.

The following is a checklist to use when troubleshooting the `gsi-interface-status` attribute of a `gsi-interface`:

- 1 The ping manager executable is started up on a Unix window or Windows Command prompt. No error messages should appear at the startup of the `pingmgr` executable. If the startup is successful, the executable should output the line stating that it is *Waiting to accept a connection*.
- 2 Check for error messages on the machine that is running the `pingmgr`. If you have opened a Unix window or Windows command prompt for the executable command, look there for execution errors. If you have redirected the output to a log file via a `>` or `>>` in the executable command, then look in that log file. Check for any suspended execution or repeated error messages for Ping Manager.
- 3 If a -2 has been detected on any of the `gsi-interfaces` of IPRA (`IPRA-SCHEDULED-PING-INTERFACE`, `IPRA-DEMAND-PING-INTERFACE`, `IPRA-SYNCHRONOUS-INTERFACE`, `IPRA-EVENT-INTERFACE`) rules will fire to reset the IPRA `gsi-interfaces` via the `ipra-reset-interface`. You can accomplish this manually by disabling and then enabling the `gsi-interface`.
- 4 Make sure that the `Gsi-interface` is enabled and not disabled.
- 5 When a network connection is broken, or a host processor is not available, the ping manager executable may have stopped. Check for error messages and/or suspension of the executable, using step 1, above. If the executable has stopped, then it should be restarted.
- 6 In G2, ensure that the `gsi-interface` for the `pingmgr` has its attribute `GSI connection-configuration` set up to point to the machine on which the executable is running. If the executable is running on the same machine, use `localhost`, otherwise use the name of the machine on which the executable is running. You can also use the ip address for the name.
- 7 Make sure that you can ping the machines from each other. If the executable is running on ABC and G2 is running on XYZ, then you should be able to successfully "`ping ABC`" from a command line in XYZ, and "`ping XYZ`" from a command line in ABC.
- 8 Make sure that the port-number is not in conflict with another `gsi-interface` port-number. Use the G2 inspect: "`display a table of every gsi-interface`".

Summary of IPRA Default Behavior

Two key IPRA classes are:

- `ipra-snmp-device`

Describes objects that have an SNMP agent installed. This implies that AutoDiscovery will have obtained information regarding the interfaces of this device. The assumption is made that this device is in some way important to the network and therefore merits some form of monitoring. The default behaviour of IPRA is to therefore set this device as a MANAGED device, meaning that its pollable interfaces will be schedule polled (see below for an explanation of pollable and scheduled polling). It is intended that this class hierarchy be extendable by the user or by future releases of IPRA to contain previously unknown types of SNMP devices.

- `ipra-packet-wrangler`

Describes objects that have packet routing capabilities. This includes but is not limited to: routers, switches, switch-routers. `ipra-packet-wrangler` is a subclass of `ipra-snmp-device`. It is intended that this class hierarchy be extendable by the user or by future releases of IPRA to contain previously unknown types of routing devices.

It is imperative that the user place new class definitions in the correct hierarchy. All non-packet-routing SNMP devices inherit from `ipra-snmp-device` and all packet-routing SNMP devices inherit from `ipra-packet-wrangler`.

IPRA has the following default behavior:

- All SNMP devices are managed.
- All non-SNMP devices are unmanaged.
- All interfaces of devices that inherit from `ipra-packet-wrangler` are non-terminal.
- All interfaces of devices that do not inherit from `ipra-packet-wrangler` are terminal.
- An interface is considered pollable if:
 - It is `dxl3-connected-to` another interface.
 - It is not an ISDN or software-loop-back interface.
- All interfaces that belong to a managed device that are pollable are also managed.
- All managed interfaces are schedule polled.

Note that the `ipra-ora-two-poll-node` method takes care of which poll to perform:

- SNMP if the `osi-layer3-address` is empty.
- ICMP if the `osi-layer3-address` is not empty.

When `ipra-ora-two-collect-x-nodes` is called (where `x` is terminal/non-terminal/passive), it checks to see that the node is of type `x` and if it is pollable. This accounts for interfaces which have been discovered but are not pollable because they are not connected to another interface.

Procedures

IPRA procedures are user customizable routines for initializing, checking `ipra-interface` objects, and setting the `gsi-interfaces` used for polling.

`ipra-scheduled-poll-initialization`

(*pi*: class `gsi-interface`)

Initializes the interface and scheduled polls. The user should make changes if necessary to any of its called procedures. This initialization does the following, by default:

- Sets the management-state of SNMP Devices by calling `ipra-set-snmp-management-status`.
- Decides if an interface is pollable by calling `ipra-check-object-pollable`.
- Decides if an interface is managed by calling `ipra-check-object-managed`.
- Creates a ping configuration file of all managed interfaces that have `osi-layer-3-addresses` and loads it into the Ping Manager bridge by calling `ipra-set-interfaces-to-icmp-poll`. This procedure assumes G2 and the Ping Manager bridge are on same machine.
- Starts scheduled polling of managed interfaces that have no `osi-layer-3-address`.

By default this procedure sets all SNMP devices as managed objects - if desired override the for-loop to alter the selection of managed/unmanaged SNMP devices. Note that this directly affects the number of interfaces that will be schedule polled because only those interfaces whose containing device is managed will be collected for scheduled polling (both ICMP and SNMP).

ipra-set-snmp-device-management-status

()

Checks if an object is an IPRA pollable interface. The user can extend the number of unpollable interfaces by editing this procedure, at a minimum the following defaults should be adhered to:

- The interface should not be a software loopback
- The interface should not be ISDN
- The interface should be dxi3-connected-to another interface

ipra-check-object-pollable

(*intf*: class ipra-interface)

-> *result*: truth-value

Checks if an ipra-interface is network pollable.

ipra-check-network-cnx

(*intf*: class ipra-interface)

-> *result*: truth-value

Checks if an ipra-interface is network connected.

ipra-check-object-managed

(*intf*: class ipra-interface)

Checks if an ipra-interface is managed. This procedure sets the ipra-interface object-management-state to either managed or unmanaged. For an interface to be managed:

- It must be pollable.
- Its superior device must be managed.

ipra-set-interfaces-to-icmp-poll

(*pi*: class gsi-interface)

Collects the interfaces that are managed and ICMP ping pollable, and writes them to a ping configuration file. It then uploads the scheduled ping gsi-interface automatically. This assumes that the G2 process and the scheduled ping manager are on the same machine. The user should make changes if this is not the case.

ipra-firewall-cnx

(*fwdevice*: text, *fwdeviceip*: text, *nwdevice*: text, *pingupdate*: truth-value)

Creates and connects devices that are behind firewall(s) to the devices in the imported domain map. You specify the *fwdevice*, which is the **opfo-external-name** of the device to which to connect, *fwdeviceip*, which is the IP address of the device to which to connect, and the *nwdevice*, which is the **opfo-external-name** of the device to which to connect. The device is created, placed on the domain map workspace, and connected to an ethernet interface of the

network device, which must be managed. If *pingupdate* is **true**, the device is added to the scheduled list (file) of polled devices and to the Ping Manager currently doing scheduled polling.

ipra-reset-domain-objects-nodes

()

Initializes all **ora-two** list attributes of ipra-interfaces. You typically call this procedure with ODG Discovery (OpEx Discovery Gateway). You must call this procedure if the map changes in any way, because the **ora-two** lists of the interfaces must be repopulated.

ipra-reset-interface

(*pi*: class gsi-interface)

Resets a **gsi-interface** if its bridge connection is lost after 60 seconds.

Object Reachability Analysis (ORA-TWO)

Provides an overview of the Object Reachability Analyzer (ORA-TWO) product, its setup, and API.

Introduction	346
Concepts	346
Setup	347
Manager Object	348
Event Methods	350
Domain Methods	351
Support Procedures	354
Additional Procedures	356
Report Procedures	357



Introduction

The Object Reachability Analyzer (ORA-TWO) product provides root cause reachability analysis for any 'threaded' network. A threaded network is one in which a physical or logical path propagates content from one object of interest to another. The content passed may be any form of concept that follows a given path, for examples:

- Data, as in the case of communications, or
- Fluid, in the case of a pipeline; or
- Electric as in the case of a power transmission grid.

Reachability analysis determines where along the path the initial cause or probable cause of the fault, and the objects effected by that failure, are located. ORA-TWO assumes bi-directional flow of content, and any successful path to an object provides reachability. ORA-TWO supports multi-path, or looping domain configurations.

Concepts

The concepts used in determining failures within a network depend upon the representation of the network and the objects contained within the network. A network or domain, in this reference, is any collection of objects tied together in some consistent form.

Within G2, connectivity, hierarchy, or relationships may tie together groups of objects. The objects themselves are referred to as nodes within a network. A node represents some part of the network that has significance to the domain.

Node Types

Three distinct types of nodes can exist within a domain:

- **Non-terminal nodes** imply that content flows through the node with no concept of direction. Multiple links to other nodes from a non-terminal device are supported in a bi-directional manner. This type of node has the ability to be tested (polled) to determine the health of the entity.

For example, a local substation for electrical power may take power or distribute power from/to other substations within its local network.

- **Terminal nodes** imply that content may flow into but not exit to other nodes; they, however, have a means of testing for health.

For example, a house connected to a local substation is a terminal node as long as a power failure monitor is available for testing.

- **Passive nodes** are those that cannot be tested. They act as terminal or non-terminal nodes but cannot be tested for current state. No direct failure events may be posted against these devices, although implied possible failures may be posted.

For example, a transformer is used to step down the voltage from the power station to the house, but the transformer cannot be tested remotely.

Classes, relationships, or other criteria may be used to determine the node type within the domain. The determination must be consistent from all views (i.e., a node may not be seen as terminal from one viewpoint and non-terminal from another).

Polling

Polling is a process to determine the state of health of the node that is polled; it is followed by a series of polls to the surrounding nodes in order to determine the state of health of the nearby domain. The initiation of network analysis is called by the user, either to `ora-two-fail-method` or `ora-two-recover-method`. These methods then begin a series of polls to the surrounding nodes in order to determine the state of health of the nearby domain. The demand polling of the local area is done in an asynchronous manner to minimize the time to determine the root cause of the failure.

Setup

The `ora-two.kb` module contains the underlying algorithm driving the analysis.

The following instructions are for merging the ORA-TWO product into an existing application.

Note If you are using Integrity, `ora-two` is already merged into your application.

To merge `ora-two` into an existing application:

- 1 Choose File > KB Modules > Merge KB and enter `ora-two.kb` to merge it into the existing application.
- 2 Display the module hierarchy by choosing Tools > Inspect and then entering the following Inspect command:


```
show on a workspace the module hierarchy
```
- 3 Specify `ora-two` as the required module by selecting the top most application module table and entering `ora-two` in the `directly-required-modules` attribute.
- 4 Transfer the methods defined within `ora-two-example.kb` to an appropriate location within your application module.

Manager Object

The ora-two-top-level workspace contains an ora-two-manager-object used for creating the ora-two manger object within the user domain.

To create a manager object:

- 1 Display the ora-two-top-level workspace, select the manger object to attach it to the mouse, and drop the manager object onto a user workspace near the object that closely represents the location of G2 within the environment.
- 2 Name the manager object and attach it to the root node of the domain (the root node supports the polling of devices).

The ora-two-manager-object is a configuration object that is attached to a node. It contains the basic configuration for messages that are generated against the node for which these messages are attached.

You should modify the table of the new ora-two-manager-object to match the current knowledge base. The two attributes of the manger object are:

Attribute	Description
ora-two-poll-complete-test-interval	The delay between checks to determine if poll is complete. (Wait for poll to complete checking every ora-two-poll-complete-test-interval.
<i>Allowable values:</i>	any positive float
<i>Default value:</i>	1.0
ora-tow-recover-poll-predicted fail-nodes	The flag to indicate if predicted to fail objects will be polled to determine health when the root cause of the predicted fail is recovered.
<i>Allowable values:</i>	any truth value
<i>Default value:</i>	True

ORA-TWO makes asynchronous polls to surrounding nodes whenever the API for either ora-two-fail-method or ora-two-recover-method is called by the user. These proactive polls are independent and may be running concurrently in different parts of the network.

Take care to filter noisy fail/recover messages to avoid system overloading due to polling. If multiple polling nodes or domains exist, create an `ora-two-manager-object` for each polling node or domain.

Event Methods

The following two methods begin propagation through the domain:

- `ora-two-fail-method` – Called when a failure event is received.
- `ora-two-recover-method` – Called when the node recovers.

These methods may be called from either a node down/up trap in the case of SNMP or poll failure/recover when using the Ping Manager. You are responsible for calling these API's from whatever generates an asynchronous event within the domain.

Each method requires the `ora-two-manager-object` associated with the object and the object itself. Multiple domains within the same knowledge base may have different manager objects defining the categories and priorities. You are responsible for determining the proper manager object for each domain object passed into the method.

The signature of these methods is:

`ora-two-fail-method`

(*omo*: class ora-two-manager-object, *tgt*: class opfo-domain-object)

`ora-two-recover-method`

(*omo*: class ora-two-manager-object, *tgt*: class opfo-domain-object)

Domain Methods

The interaction between a node and the surrounding environment requires methods that determine the:

- Type of devices within the immediate vicinity.
- Health of those devices.
- Message text to be posted against an object.

These methods are provided by the example KB and may be extended by the user depending upon the domain.

ora-two-poll-node

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)
-> status: symbol

Polls a target node of the specified manager. Returns the symbol OK or the symbol Fail depending upon the poll response from a node. The discovery of failed and recovered nodes by ORA-TWO causes the `poll-node` method to be executed for nodes around the target. This proactive polling is asynchronous and autonomous between fail and recover events.

It is the user's responsibility to ensure that:

- This method operates in a non-blocking manner, i.e. multiple polls may run concurrently.
- Multiple poll requests to the same node are handled efficiently by the polling method; for example, if three ping requests are made to the same device, the method will make only one ping and return that response to all three requests.

ora-two-poll-recover-method

(*tgt*: class object, *omo*: class ora-two-manager-object)
-> main-text: text, additional-text: text

Polls a target node of the specified manager, and returns the text and additional text used in generating a recover poll message.

ora-two-node-type

(*omo*: class ora-two-manager-object, *tgt*: class opfo-managed-object)
-> type: symbol

Determines the node type of a target node of the specified manager, and returns one of the following symbols: `terminal-node`, `non-terminal-node`, or `passive-node`. A terminal node is defined as one that will not lead to other nodes, Non-terminal nodes may lead to other nodes, and passive nodes are ignored by the analysis. Nodes may be identified by class, connections, relations, or placement.

ora-two-collect-non-terminal-nodes

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)

-> *node-list*: class item-list

Returns an item-list of all non-terminal nodes associated with a target node associated with the specified manager. The item-list is deleted when processing is complete.

ora-two-collect-terminal-nodes

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)

-> *node-list*: class item-list

Returns an item-list of all terminal nodes associated with a target node associated with a manager. The item-list is deleted when processing is complete.

ora-two-collect-passive-nodes

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)

-> *node-list*: class item-list

Returns an item-list of all passive nodes associated with a target node associated with the specified manager. The item-list is deleted when processing is complete.

ora-two-root-cause-message

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)

-> *main-text*: text, *additional-text*: text

Returns the text and additional text used in generating root cause messages for a target node associated with the specified manager.

ora-two-poll-fail-message

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)

-> *main-text*: text, *additional-text*: text

Returns the text and additional text used in generating a poll fail message for a target node associated with the specified manager.

ora-two-predicted-poll-fail-message

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)

-> *main-text*: text, *additional-text*: text

Returns the text and additional text used in generating a predicted poll fail message for a target node associated with the specified manager.

ora-two-domain-consistency-check

(*omo*: class ora-two-manager-object)

-> *node-list*: class item-list

Returns a list of lists containing related nodes associated with the specified manager. Each list contains objects that are related together by ora-two methods but are not related to the manager object. Only clusters with more than one member are included. Note that this routine traverses the entire

domain related to the manager object and may cause heavy system loading. You are responsible for deleting the returned list of lists when processing is complete.

ora-two-recursive-collect-non-terminal-nodes

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)
-> *node-list*: class item-list

Returns a list of non-terminal nodes bypassing passive nodes for a target node associated with the specified manager. The health of the node is not considered as a condition for addition to this list. You are responsible for deleting the list when processing is complete.

ora-two-recursive-collect-terminal-nodes

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)
-> *node-list*: class item-list

Returns a list of terminal nodes bypassing passive nodes for a target node associated with the specified manager. The health of the node is not considered as a condition for addition to this list. You are responsible for deleting the list when processing is complete.

ora-two-recursive-collect-passive-nodes

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)
-> *node-list*: class item-list

Returns a list of passive-nodes for a target node associated with the specified manager. The health of the node is not considered as a condition for addition to this list. You are responsible for deleting the list when processing is complete.

Support Procedures

The following procedures determine root cause nodes for a failure and nodes affected by a root failure.

Note Because failure processing is asynchronous, root cause determination and affected nodes routines will not be reliable if called from a single node message. For example, a root cause failure may be flagged before all predicted failures are determined. Conversely, a node may be determined as predicted to fail before the root cause is known.

ora-two-find-down-nodes-for-root-cause

(*tgt*: class object, *omo*: class ora-two-manager-object)
-> node-list: class item-list

Returns a list of objects that were down nodes associated with a root cause message. The returned list contains all the nodes that are predicted to fail a poll, those that have failed a poll, and all of the root cause alarms surrounding those nodes. This routine determines all down nodes known at the time of the call. The user is responsible for deleting the returned list when processing is complete.

ora-two-find-root-causes-for-manager

(*omo*: class ora-two-manager-object)
-> node-list: class item-list

Returns a list of objects that were the root cause of the failure or that predicted failure of this manager object. These are the root causes found that are associated with the manager object. This routine determines all root cause failures known at the time of the call. The user is responsible for deleting the returned list when processing is complete.

ora-two-find-root-causes-for-object

(*tgt*: class object, *omo*: class ora-two-manager-object)
-> node-list: class item-list

Returns a list of objects that were the root cause of the failure or that predicted failure of this node object. These are the root causes found that are associated with any object. This routine determines all root cause failures known at the time of the call. The user is responsible for deleting the returned list when processing is complete.

ora-two-recursive-collect-non-terminal-nodes

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)
-> node-list: class item-list

Returns a list of non-terminal nodes, bypassing passive-nodes. The health of the node is not considered as a condition for the addition to the returned list. You are responsible for deleting the list when processing is complete.

ora-two-recursive-collect-terminal-nodes

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)

-> *node-list*: class item-list

Returns a list of terminal nodes, bypassing passive-nodes. The health of the node is not considered as a condition for the addition to the returned list. You are responsible for deleting the list when processing is complete.

ora-two-recursive-collect-passive-nodes

(*tgt*: class opfo-managed-object, *omo*: class ora-two-manager-object)

-> *node-list*: class item-list

Returns a list of passive nodes. The health of the node is not considered as a condition for the addition to the returned list. You are responsible for deleting the list when processing is complete.

Additional Procedures

ora-two-get-state

(*tgt*: class object, *omo*: class ora-two-manager-object)

-> *state*: symbol

Returns one of the following symbols that represents the reachability state of the object:

- good – State is assumed to be good.
- predicted-fail – This object is predicted to fail poll.
- poll-fail – This object has failed a poll.
- root-cause – Root cause failure.

ora-two-release-object-states

(*omo*: class ora-two-manager-object)

Releases any objects that have registered ORA-TWO relations with the manager object by deleting the relationships. Additionally, this procedure empties internal reference lists and deletes unused poll objects. Note that this is done on a per manager basis.

Report Procedures

`ipra-ora-two-report`

(tgt: class ipra-domain-object, win: class g2-window)

-> *report: text*

Returns a text string containing a list of all terminal and non-terminal nodes in the network domain along with their external names (`opfo-external-name`).

Use this procedure as a debugging tool to determine whether the network has been set up properly.

`ipra-ora-two-report`

(tgt: class ipra-domain-object, win: class g2-window)

-> *report: text*

Returns a text string containing a list of all terminal and non-terminal nodes in the network domain along with their external names (`opfo-external-name`).

Unlike `ipra-ora-two-report`, this method provides an abbreviated list of nodes without any repeats.

Domain Export/Import (DXI3)

Provides a description of how to import and update the domain map.

Introduction **360**

Integrity Export Import Toolbox **360**

The DXI3-file Format **360**

A “Bad” Import File and Data Corruption **363**

Type to Class Mapping **364**

Containment and Other Types of Hierarchies **365**

Exporting Domain Maps **365**

Importing a Domain Map **368**

Example **369**

DXI3 APIs **372**

 dxi3-register-domain-item **373**

 dxi3-register-domain-relation **374**

 dxi3-register-domain-attribute-value **375**



Introduction

The Domain eXport Import (DXI3) component allows you to import and export domain objects into Integrity.

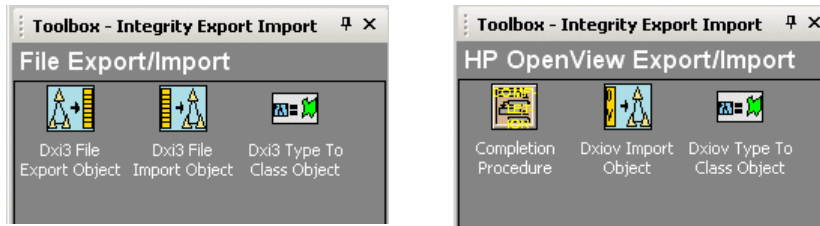
The two ways of importing or updating the domain map in Integrity are:

- Calling APIs that are defined by the DXI3 module.
- Reading a text file formatted in the dxi3 file format defined by the DXI3-FILE module.

The DXI3-FILE module is a shell around the DXI3 module. It parses the import file and calls the DXI3 APIs. Understanding the behavior of the dxi3 file import, requires an understanding of the DXI3 APIs.

Integrity Export Import Toolbox

Here are the palettes in the Integrity Export Import toolbox:



The DXI3-file Format

The format for the dxi3 import file is the following:

*****<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>	(Nx optional)

*****<object id>	(required)
relation: <relation name> [<object id>,...]	(Nx optional)
delete-relation: <relation name> [<object id>,...]	(Nx optional + required empty line at end)

The dxi3 import searches for an empty line followed by the 6 stars and the object-id.

Every line after this line is optional and can appear in any order. Each line represents a single call to any of the three register-domain APIs in DXI3. The 'descriptor' at the start of each line determines which of the three is called:

- 'action:' implies call `dxi3-register-domain-item(DXI, <object id>, <type>, {'create' | 'delete'})`
- 'superior:' implies call `dxi3-register-domain-relation(DXI, <object id>, 'superior', <object id>, 'create')`.
- 'delete-superior:' implies call `dxi3-register-domain-relation(DXI, <object id>, 'superior', <object id>, 'delete')`.
- 'connected:' implies call `dxi3-register-domain-relation(DXI, <object id>, 'connected', <object id>, 'create')`.
- 'delete-connected:' implies call `dxi3-register-domain-relation(DXI, <object id>, 'connected', <object id>, 'delete')`.
- 'attribute:' implies call `dxi3-register-domain-attribute(DXI, <object id>, <attribute name>, <attribute value>)`.
- 'relation:' implies call `dxi3-register-domain-relation(DXI, <object id>, <relation name>, <object id>, 'create')`.
- 'delete-relation:' implies call `dxi3-register-domain-relation(DXI, <object id>, <relation name>, <object id>, 'delete')`.

Remarks on the Syntax

- The dxi3-file must contain a nonempty header. The dxi3 import searches for the header, then searches for an empty line followed by the 6 stars and the object-id. Every line afterwards that is within the object-definition is optional and can appear in any order. The object definition ends with the next empty line.
- The SPACE character is NOT ignored unless it is explicitly used as a separator. For this reason, object-ids and types can be strings that include spaces.

- The list of object-ids in the 'connected:', 'delete-connected:', 'relation:' and 'delete-relation:' lines should be separated by a comma only. Any space before or after the comma is considered to be a part of the object-id.
- The 'action:' line does not require a type. In case of delete, a type is ignored. In case of create, if the type is an empty string, the default class is used (a dxi3 feature).
- If no 'action:' line is specified, the object must exist and is modified by the lines (a dxi3 feature) that follow.
- The 'superior:', 'delete-superior:', 'connected:', 'delete-connected:', 'relation:' and 'delete-relation:' can have no object-id(s) specified and these lines are effectively ignored.
- In 'relation:' and 'delete-relation:' the relation-name should be at least one character and not a space. Also, the relation-name should be followed by a single space, otherwise the line is ignored.
- If the descriptor (i.e. any text between the beginning of the line and the first '!') is unknown, an exception will be reported and the line will be ignored.
- If the action is not 'create' or 'delete', an exception is reported and the line is ignored.
- If the attribute name cannot be parsed correctly (i.e. the attribute name is an empty string or the delimiter '=' does not exist), an exception is reported and the line is ignored. The attribute value is allowed to be an empty string (and thus to have no characters).
- If the relation name cannot be parsed correctly (for example, the relation name is an empty string or the delimiter ' ' does not exist), an exception is reported and the line is ignored. The list of object-ids is allowed to be an empty list (and thus to have no characters).
- The user-defined relations should be read in 'reverse order'. For example:

```
*****IPAddress1
```

```
relation: the-subnet-of SubnetA'
```

Should be read as: SubnetA will be 'the-subnet-of' IPAddress1.

From dxi3 release 3.1:

- Apart from the first carriage return (which indicates the end of the header), all other carriage returns are ignored. As a result, the file reader is insensitive to wrongly placed carriage returns.
- A line that starts with a comment sign ('#') is ignored. Note that for compatibility reasons, all lines up to the first carriage return are ignored, regardless of whether the line starts with the comment sign.

- The 'six stars' containing the object-id effectively sets a local variable containing the current object-id. All other lines following this statement will result in dxi3 api-calls using this object-id. It is the responsibility of the dxi3 file generator to ensure that the first statement after the header sets the object-id. Otherwise, the object-id 'no-name' will be used.

A “Bad” Import File and Data Corruption

Under certain circumstances, the import dxi3 file or calls to the dxi3 APIs might contain bad data. (A discussion of the definition of bad follows.) Such bad data has an impact on the domain map in Integrity. This section discusses the types of bad data that can be recognized, the way the dxi3 import handles the types of bad data and the effects of it on the domain map in G2.

Types and Handling of 'Bad' Data and DXI3

The following lists the different types of bad data that the dxi3 import may contain. This list is divided into two parts: errors particularly related to the dxi3 import file, and errors that are common between the dxi3 API and file-import use.

Errors Particular to the dxi3-import File

Errors that apply to the dxi3-import file are:

- Wrong use of delimiters between the object definitions (i.e. other than an empty line followed by 6 stars and the object-id). Result: import will stop.
- Use of an undefined descriptor. Result: line will be ignored.
- Wrong use of delimiters within a line. Result: if the error in the delimiters is recognizable, the line will be ignored. Otherwise, inevitably delimiters will be parsed as part of the contents (e.g. attribute name, object-id, etc.).

In summary, these errors are formatting errors. Any automatic dxi3 file generation process should be easily debugged such that it will generate a dxi3 file without these errors.

Errors Common to the dxi3 API and File Use

Errors that are common to the dxi3 API and file use are:

- Use of an object-id of an object that does not exist or will not be generated within the same commit or that will be deleted within the same commit. Result: that particular change in a relation or attribute will be ignored. Two example causes of this type of error are:
 - Inconsistent generation of object-ids (i.e. object-id within create line is generated differently then in a relation or attribute API call). This is a formatting error and should be easy to debug in the generation process.

- Reference to an object outside the set included in the import. This is a problem of the database query and filter. By ignoring the reference, the dxi3 import will effectively filter the object.
- Use of an attribute or relation name that is not defined for the class of object(s) is requested for. Or, the use of an attribute value that does not fit the class definition. Result: the API call will be ignored. This indicates that the class and relation definitions in Integrity are not consistent with the expectations in the generation process. This should be moderately easy to debug.

Effects of 'Bad' Data on the Domain Map

Note Even if bad data enters the dxi3 import process, the domain map in G2 will never be corrupted from an Integrity perspective. A domain map that results from an import that contains “bad” data, from an Integrity perspective, will be a valid domain map; Integrity will not crash because of it.

The only effect of bad data on the domain map is an increase of the differences between the domain map and the physical network that it represents.

However, many other reasons may exist for differences between the Integrity domain map and the physical network that it represents in addition to the exceptions that the dxi3 import process is able to recognize. If the process that generates the dxi3 file or that is calling the dxi3 APIs is properly debugged, the number of recognizable exceptions in the dxi3 import will be small. After proper debugging, the only resulting inconsistencies in the dxi3 import are those that are very difficult or impossible to detect.

Type to Class Mapping

The type as specified by the `dxi3-register-domain-item` system procedure is mapped to a G2 class definition as follows:

- 1 If a `dxi3-type-to-class` object (upon the translation workspace) that has an exact mapping to a `g2` class exists, this object is used first.
- 2 Next, if a `dxi3-type-to-class` object exists that matches the type using a regular expression, this object is used.
- 3 If no matching translations exist, `dxi3` looks for an exact mapping to a `g2` class name.
- 4 If an exact `g2` class name is not found, the default class is used.

A `dxi3-type-to-class-object` can be cloned from the palette upon the top-level workspace of `dxi3`.

Containment and Other Types of Hierarchies

In contrast to the dxi(1) import, dxi3 subordinate objects are related to superior objects via a g2-relation instead of g2-connections. DXI3 builds the hierarchy of objects via relations at import. For visual organization of the imported objects, DXI3 places subordinate objects upon the subworkspace of a superior object. DXI3 depends on the g2-relations for hierarchy operations, not on the placement of the objects.

As a consequence, the dxi3 imported domain map is no longer intended as a user-interface. Also, it is not intended to manually modify the domain map.

The reasons for these design decisions are:

- 1 Many different hierarchical views of the network/domain map are possible. Users may want to view the same domain map in multiple ways. For instance, a user may want a view based upon a geographical hierarchy, or based upon a ip/subnet hierarchy, or based upon a component class hierarchy. Gensym is currently developing proxy views that can act as a user-interface for the domain map.
- 2 The dxi3 import is incremental and allows for deletion of objects. If an object is deleted in G2, all components on its subworkspace are also deleted. Thus, from a dxi3 standpoint, objects may be deleted that are not explicitly stated. Since the hierarchy is very subjective, deletion might cause confusing results. For instance, using the dxi3 APIs, a user might want to create new subnets and delete the old ones. If the routers assigned to a subnet have been placed upon the subworkspace of the subnet, all these routers will be deleted if the subnet is deleted. By not placing the routers upon the subworkspace, only the subnet is deleted and all its g2-relations to the routers. The next API calls might create a new subnet and create new relations to the existing routers. Conclusion: to delete an object in dxi3 delete the object explicitly.
- 3 Expressions using g2-relations are much faster to evaluate than those using g2-connections.

Notes:

- Multiple dxi3-type-to-class definitions may map to the same g2 class.
- Dxi3-type-to-class-objects are NOT used for file export.

Exporting Domain Maps

The domain-map can only be exported to a dxi3 export file using the dxi3-file module. DXI3 does not contain particular APIs for exporting.

The dxi3 file exports the following data:

- All opfo-domain-objects upon the domain-map workspace. (Note that any non-opfo-domain-object are excluded.)
- The object-id is the opfo-external-name of the object.
- The type is the g2-class name of the object.
- A superior line is only generated if a superior object exists.
- All connected objects.
- All attributes that are of the type `value` whose names start with any of the prefixes (as specified in the export-object) and that are not hidden opfo-domain-object attributes.
- All relations between domain-map objects that are not the standard dxi3 relations. (For example, standard relations would be `superior` or `connected`.)

The dxi3-file exporter generates a dxi3-file that can be imported and results in exactly the same domain map. However, this similarity does not mean that the dxi3-export file will be identical to the dxi3 file used to import the domain map. The export file could be different in the following ways:

- The ordering of the object definitions.
- Objects are only be created and never deleted.
- The type of the object. The type is always the name of the g2-class of the domain item. This means that the dxi3 file export does NOT use dxi3-type-to-class-objects. The reason is that the type-to-class mapping a multiple to one mapping is during import. For export, this would result in a one-to-multiple mapping and is thus infeasible.
- All relations are listed double. That means, if object A is-related-to object B, the definition of object A would contain the line `relation is-related-to B` and the object definition of object B would contain the line `relation is-related-to A`. During import, each of these lines alone will set the relation.
- The dxi3 export exports all user-defined relations between objects within the same domain map, excluding the standard dxi3 relations (i.e., `superior` and `connected`).
- The dxi3 export exports all attributes that start with any of the specified prefixes.

To export a domain map:

- 1 Create an instance of a `dxi3-file-export-object`, by cloning it from the Toolbox - Integrity > Export Import palette. Specify the following attributes:
 - a `dxi3-file-name`: the full path and file name of the dxi3 export file.
 - b `dxi3-domain-map-workspace`: The name of a workspace or the name of an object with a subworkspace that will hold the instances of the domain map.
 - c `dxi3-output-prefixes`: a sequence of text-strings. The text-strings should represent prefixes of attribute names to export. The idea is that not all attributes of an object need to be exported. By starting with a particular prefix the names of those attributes that need to be exported, the attributes can be easily selected. By providing a sequence, multiple prefixes can be specified. If an attribute name starts with any of these prefixes, it will be exported. The default value is sequence(""), which implies that all user-defined attributes will be exported. The effect of specifying an empty sequence `sequence()` is that no attributes are exported.
- 2 Choose **Start File Export** to start the export process. Next to the import object, the attribute `dxi3-status` is displayed. This status shows the actual status of the export process and any exceptions found.
- 3 The most important status messages and exceptions are logged in the hidden attribute `_dxi3-notes-list`. This attribute can be viewed as administrator. In addition, there is a hidden attribute `dxi3-exception-count` that counts all exceptions.
- 4 The hidden attribute `_dxi3-output-attributes` can be viewed as administrator and can be used for debugging. It is an intermediate result, and shows the classes and their attributes that will be exported. (Note that the dxi3 export does not and cannot use `dxi3-type-to-class-translations` since this is not a one-to-one mapping. Instead it exports the `g2-class-name` as the type.)

In addition to the actual export of a dxi3 file, the export object can also be used to generate two types of reports that help you analyze the existing domain map:

- **report containment hierarchy** writes a text-file that describes the containment hierarchy and the connections. The path and name of this file is `dxi3-file-name-containment-report`.
- **report class hierarchy** writes a text-file that lists all the objects in the domain map and their class, sorted by class and name. The path and name of this file is `dxi3-file-name-class-report`.

Importing a Domain Map

To import a domain map:

- 1 Determine the classes of objects to import, the object attributes to import and the type of user-defined relations between these classes to import.
- 2 Define these classes and relations in G2. (If you ever want to export a dxi3 file, make sure the name of the attributes you want to export start with a common prefix. See [Exporting Domain Maps](#).)
- 3 If required, define dxi3-type-to-class-objects that map the type as used in the dxi3 api-call or file to the correct G2 class name. The use of dxi3-type-to-class-objects can be avoided by using the G2 class name as the type. See [Type to Class Mapping](#).
- 4 Determine a consistent naming convention for the object-ids. The DXI3 import requires only that object-ids must be unique. However, beware that when traps arrive, a completion procedure must be able to determine the target of the trap by generating the object-id out of the information in the trap. Thus, ensure that the object-id can be constructed out of names that arrive within the data of a trap.
- 5 Generate a correct dxi3 import file.
- 6 Create an instance of a `dxi3-file-import-object`, by cloning it from the palette on the `dxi3-file-top-level` workspace. Specify the following attributes:
 - a `dxi3-translation-workspace`: The name of a workspace or the name of an object with a subworkspace that holds the optional dxi3-type-to-class-objects. If no translations are required, specify `no-translations`.
 - b `dxi3-default-class-to-create`: a g2 class name (by default `opfo-domain-object`). If the type of an object cannot be mapped to a G2 class, this default class is used.
 - c `dxi3-domain-map-workspace`: The name of a workspace or the name of an object with a subworkspace that will hold the instances of the domain map. This workspace will hold all instances of the domain map. Therefore subordinate objects will NOT be placed upon the subworkspace of a superior object. See [Containment and Other Types of Hierarchies](#). Since the DXI3 import is incremental, existing objects will be modified. In case of a completely new import, delete all old domain objects upon this workspace.
 - d `dxi3-release-after-commit`: This is a debugging feature. The dxi3 import process generates intermediate registration-objects. Being able to look at these objects can be useful for debugging purposes. The default of this attribute is `true`, which implies that these intermediate objects will be deleted at the end of the import. If this attribute is set to `false`, the objects

are retained. The next start-import will automatically delete them before the actual import process starts.

- e **dxl3-only-consider-opfo-domain-objects.** This is an optimization feature. If you are sure that you are only importing subclasses of opfo-domain-object (as is the case for any regular domain-map import), setting this flag to true optimizes the performance of the import.
 - f **'dxl3-do-not-check-for-multiple-registrations.** This is an optimization feature that can improve performance significantly in case of very large domain maps. Any inconsistent item registration will be detected anyway by other exception handlers.
 - g **dxl3-default-column-height, dxl3-vert-spacing, and dxl3-horz-spacing:** the dxl3 import places new objects upon the specified domain map workspace in columns. These three attributes define, respectively, the height of these columns, and the vertical and horizontal spacing between the objects.
 - h **dxl3-file-name:** the full path and file name of the dxl3 import file.
- 7 Choose **Start File Import** on the dxl3-file-import-object to start the import process. Next to the import object, the attribute **dxl3-status** is displayed. This status shows the actual status of the import process and any exceptions that it finds.
 - 8 The most important status messages and exceptions are logged in the (hidden) attribute **_dxl3-notes-list**. This attribute can be viewed as administrator. In addition, there is a (hidden) attribute **dxl3-exception-count** that counts all exceptions. The exception list can be viewed in two ways:
 - a By describing the notes list.
 - b By choosing **Write Exceptions As File**. This selection writes the exceptions list as a file called *dxl3-file-name-exceptions*.
 - 9 If no exceptions occurred or if the exceptions appear to be acceptable, you are ready to use the domain map.

Example

This example illustrates the use of dxl3. It is a compact example of a hypothetical and not necessarily realistic network. This example network contains a variety of components. The dxl3 import file illustrates the use of many features of dxl3.

The example requires the following files:

- *dxl3-example.kb*
- *dxl3-example.txt*

Loading *dxl3-example.kb* loads the dxi and dxi-file modules. It contains all necessary class definitions and workspaces.

To test the dxi3 import:

- 1 Ensure the path of the import file in the import object is correct.
- 2 Delete any existing domain map on the domain map workspace.
- 3 Choose Start File Import on the import object.

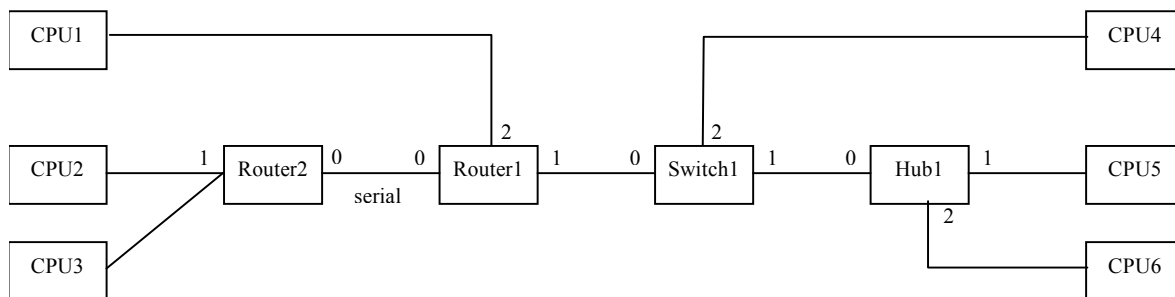
To test the dxi3 export:

- 1 Ensure the path of the import file in the import object is correct.
- 2 Choose Start File Export on the export object.

The Example Network

The following is a drawing of the network in the example. Note that the dxi3 import does NOT result in a similar visual layout. In order to verify the correctness of the import, one can either inspect the objects and relations directly in G2, or create the class and containment-hierarchy reports.

(Note the numbers next to the device-connections are the interface-indices.)



The Data Structure

The following is a description of the types of objects and their attributes and relations:

- Type: Myprefix-Network
- Type: Myprefix-Router
 - superior: <a myprefix-network>
- Type: Myprefix-IP-Card
 - superior: <a myprefix-router or a myprefix-switch>
 - attribute: myprefix-interface-index = <an integer>
 - connected: <one or more myprefix-ip-card / myprefix-switch-port / myprefix-hub-port>

- Type: Myprefix-Serial-Card
 - superior: <a myprefix-router>
 - attribute: myprefix-interface-index = <an integer>
 - connected: <one myprefix-serial-card>
- Type: Myprefix-IP-Address
 - superior: <a myprefix-ip-card / myprefix-switch-port / myprefix-hub>
 - attribute: myprefix-ip-address = <a text>
 - relation: the-subnet-of <a myprefix-subnet>
- Type: Myprefix-Subnet
 - superior: <a myprefix-network>
 - attribute: myprefix-subnet-mask = <a text>
- Type: Myprefix-Switch-Port
 - superior: <a myprefix-switch>
 - attribute: myprefix-interface-index = <an integer>
 - connected: <one or more myprefix-ip-card / myprefix-switch-port / myprefix-hub-port>
- Type: Myprefix-Hub
 - superior: <a myprefix-network>
- Type: Myprefix-Hub-Port
 - superior: <a myprefix-hub>
 - attribute: myprefix-interface-index = <an integer>
 - connected: <one or more myprefix-ip-card / myprefix-switch-port / myprefix-hub-port>

Notes/Assumptions

- A myprefix-switch might have as subordinate objects both myprefix-ip-addresses and myprefix-switch-ports.
- A myprefix-switch-port will not have an ip-address.
- A myprefix-hub may or may not have an ip-address as a subordinate.
- A myprefix-hub-port will not have an ip-address.
- When reusing parts of this example in a user's kb, the user should change the prefix 'myprefix' to any prefixes appropriate to the user's application.

1 Attributes of the superclass opfo-domain-object will never be exported.

DXI3 APIs

DXI3 provides APIs to create or destroy objects within G2, relate or unrelate the objects, and assign attribute values to the objects. Import is associated with an import object, multiple import objects may exist and operations are independent, (registrations to one import object have no impact on others).

The general sequence of calls against the import object are as follows:

- 1** `dxI3-release-domain-registrations` {deletion of any outstanding registrations and general initialization}
- 2** `dxI3-register-domain-(item / relation / attribute)` {register additions / deletions}
- 3** `dxI3-commit-domain-registrations` {apply items / relations / attributes}
- 4** `dxI3-release-domain-registrations (dxI: class dxI3-import-object) = (truth-value, text)`
- 5** Releases any registrations associated with the import object and performs initialization.
- 6** Returns a success-flag, (true = success), and a text notes string. Failure indicates some attribute of the import object is incorrect as described in the notes string.

dxi3-register-domain-item

Registers the existence of an item in the domain.

Syntax

dxi3-register-domain-item

(*dxi*: class dxi3-import-object, *node-name*: text, *type*: text,
create-or-delete: text)
 -> success: truth-value, notes: text

where:

- *dxi* is the import object.
- *node-name* is the opfo-external-name of an opfo-domain-object or the name of an item.
- *type* should match either the dxi3-type-string of a dxi3-type-to-class-object or the name of an object-definition.
- *create-or-delete* is one of these texts:
 - "create" builds the item.
 - "delete" destroys an existing item.

The order of checking is:

- 1 Check for an exact dxi3-type-to-class-object match.
- 2 Check dxi3-type-to-class-objects for the closest match by using a reg-ex expression (for example, "test-computer.*" matches closer to a type of "test-computer-small" than does 'test-computer' which is a better match than 'test'.
- 3 Test if there exists an object-definition named by type.
- 4 If none of the above checks is positive, type is the dxi3-default-class-to-create of DXI.

Returns

Returns a success-flag, (true = success), and a text notes string.

dxl3-register-domain-relation

Registers the existence of a relationship between two items.

Syntax

dxl3-register-domain-relation

(*dxl*: class dxl3-import-object, *node-1*: text, *relation-name*: text, *node-2*: text,
create-or-delete: text)

-> success: truth-value, notes: text

where:

- *dxl* is the import object.
- *node-1* and *node-2* are the opfo-external-names of opfo-domain-objects or the proper names of G2 objects.
- *relation-name* specifies the name of a G2 relation or one of two keywords:
 - superior indicates that node-1 is superior to node-2.
 - connected indicates that node-1 is connected to node-2.
- *create-or-delete* is one of these texts:
 - "create" builds the relation.
 - "delete" destroys an existing relation.

Returns

Returns a success-flag, (true = success), and a text notes string.

dxi3-register-domain-attribute-value

Registers the change of an attribute of an item.

Format

dxi3-register-domain-attribute-value

(*dxi*: class dxi3-import-object, *node-name*: text, *attribute-name*: text,
attribute-value: text)

-> class dxi3-attribute-receiver

where:

- *dxi* is the import object.
- *node-name* is the opfo-external-name of an opfo-domain-object or the name of an item.
- *attribute-name* is the name of the user-defined attribute of the item.
- *attribute-value* is the text equivalent of the value. The current type of the attribute is used to convert the text to the proper type.

Returns

Returns a success-flag, (true = success), and a text notes string.

Notes

The dxi3 format is inherently differential. If the object already exists, it will be modified accordingly.

The dxi3 commits relations effectively according to the syntax conclude that *object1* is now *relation-name object2*. That means that any conflicting relation will be deleted. For example, the superior relation is a one-to-many relation i.e., an item can only be subordinate to one other item. If, in the existing map, A is superior to B and the api concludes C is superior to B, the relation A is superior to B will be deleted. The same holds for any user-defined relation that has some restriction, e.g., one-to-many, many-to-one, one-to-one.

Open View Map Importer (OVMAP)

Provides an introduction to the Open View Map Importer (OVMAP module and describes how to install and setup OVMAP .

Introduction	377
System Requirements	378
Installation	378
Detailed Descriptions	381
OV Map Importer Operation	387
Notes on GD XI	388



Introduction

The OpenView Map Importer (OVMAP) allows the importation of the IP domain objects discovered by HP OpenView into a G2 knowledge base. The interface with OpenView is through the `ovobjprint` command-line function. This function creates an ASCII file that describes the IP devices, Nodes, Segments, and Sub-networks discovered by OpenView.

The product supports two modes of operation:

- **Batch mode:** Initially, OV Map Importer creates the IP domain within G2 as a batch process. All devices within OpenView are written to a single file; OV Map Importer then reads that file and creates the domain.
- **Incremental mode:** Secondly, it operates in an incremental mode. A “new IP device discovered” trap from OpenView is received, causing a series of

individual object file requests that allow the creation and placement of the newly discovered items (the module OXS is required for SNMP communications with OpenView).

OV Map Importer does not create the visual representation of an OpenView window; it creates only the connectivity between IP aware devices used for reachability analysis.

The OV Map Importer is referenced within the GDXI module.

Layer 2 discovery information can be obtained from NNM by using the command `ovet_topoquery getAllNodes -ShowIF` as tested with NNM 6.5. This information can also be imported into an Integrity application.

System Requirements

This module has been tested with OpenView NNM Version 6.5.

OV Map Importer works in conjunction with the module OXS (Operations eXpert Snmp) and a G2 SNMP OpenView SNMP bridge, to provide trap information for incremental creation of objects. If the knowledge base is not to include OXS, the import will create the domain only in batch mode.

The GDXI knowledge base module for OV Map Importer is a required module of IPRA (IP Reachability Analyzer). IPRA changes the loading of modules but not the testing, setup, or operation.

Installation

Network Account Setup

The GDXI OV Map Importer module communicates with OpenView through command line calls to the OpenView function `ovobjprint`. This function creates a file of the items depending upon the command line options given. In order for the function to operate properly, a user account must be setup on the machine that is running G2 as well as the machine supporting OpenView. The account must have the same name on both machines. The account must be a trusted account on both machines (your system administrator can help).

Note If G2 is running on a Windows system, the user's password on OpenView must be set to a null string ("").

Ovobjprint Command

The ovobjprint command consists of the command, options, and any redirection of output. Examples of the command:

- To list all objects to /tmp/objprint.txt:
\$ /usr/OV/bin/ovobjprint > /tmp/objprint.txt
- To list all objects to /tmp/objprint.txtj:
\$ /usr/OV/bin/ovobjprint -o 12345
- To list information about selection name serial:device1:
\$ /usr/OV/bin/ovobjprint -s serial:device1

Testing

Testing consists of checking that the:

- Command line function operates.
- Command can be performed across the network.

To check that the command line function operates:

- ➔ Login to the OpenView machine and execute the following ovobjprint command:

```
$ /usr/OV/bin/ovobjprint > /tmp/objprint.txt
```

Note Depending on the version and installation of OpenView, you may have to modify the path of the ovobjprint command (e.g. /opt/OV/bin/ovobjprint > /tmp/objprint.txt).

This command requests all object information to be saved to the file:

```
/tmp/objprint.txt
```

This command may take 1-to-25 minutes, depending on the activity on the OpenView machine. The file may be quite large (10 - 15 Mb is not unusual). After completion, open the file with any editor and verify that the file is readable and has a format similar to that described under [Translation Objects](#). After the testing you may delete the file /tmp/objprint.txt.

To execute the command across the network:

- ➔ Login to the G2 machine and execute the following command:

```
$ rsh hostname -l username -n /usr/OV/bin/ovobjprint > /tmp/objprint.txt
```

where:

`hostname` is the name or IP-address of the OpenView machine;
optionally, the `-l username` (`-l` is a lower case "L") is the trusted account name that may have been setup earlier.

Note If `rsh` does not work on your machine, you may try `remsh`.

This command should finish in approximately the same amount of time as the first test and deposit the same file in the `/tmp` directory of the local machine, or `c:\temp` on Windows.

Make a note of the address, account name, and directory of OpenView, as these will be needed to setup the [Initializations](#) of OV Map Importer.

To execute the `ovet_topoquery` command:

- 1 Login to the machine running NNM 6.5 or higher.
- 2 Setup the local environment (refer to the NNM documentation).
- 3 Change directories to the `/opt/OV/support/NM` directory.
- 4 Enter the following command to obtain all nodes and interfaces:

```
ovet_topoquery getAllNodes -ShowIF > /tmp/layer-2-info.obj
```

This will create a file called `layer-2-info.obj`, which can be FTP'd to the G2 machine and imported through the main Setup dialog under the Domain Import tab.

Installation of Modules

OV Map Importer is a part of IPRA. The installation is included in the IPRA installation. If using the product as a standalone product or if you want to merge this product with an existing one, see [IP Reachability Analyzer \(IPRA\)](#).

Setup of Incremental Addition of Domain Objects

If you are merging OV Map Importer (GDXI.KB) into your application and you would like to incrementally add domain objects into your application, you must configure GTAP.KB (Operations Expert SNMP) and a G2 SNMP OpenView Bridge.

You can configure OV Map Importer as follows to receive traps from OpenView that indicate when new IP devices have been discovered (these changes are already completed in IPRA).

To create a new trap class:

- ➔ Create the new trap class under the Receiver Objects.

Go to a current Class definition and create the new object by choosing New Definition from the KB Workspace menu, then configure the class as follows:

Attribute	Value
Name	TRAP-HPOPENVIEW-6-58785792
Direct-superior-classes	SNMP-GENERIC-TRAP
Class-specific-attributes	Add: trapd-format initially is "IF \$7 added"

To create a completion procedure:

- 1 Display the Completion Procedures workspace in your application.
- 2 Display the OV-MAP-IMPORTER-PALETTE-WORKSPACE.
- 3 Clone a Completion procedure by selecting the procedure at the bottom of the palette, and place it on your Completion Procedures workspace.
- 4 Change the name to COMPLETION-HPOPENVIEW-6-58785792.
- 5 Remove all comments in the parameter list and in the body of the procedure.

You must also perform the following setup tasks:

- Include OXS as a required module of the OV Map Importer (DXIOV).
- Configure OpenView to forward this trap to G2.

Detailed Descriptions

Class Definitions

Here are classes defined in the OV Map Importer.

dxiov-import-object

This object is used to configure a session for obtaining and parsing a file generated from OpenView to create network objects in your application.

Note When the attribute name includes 'workspace', this attribute is a symbol that points to a named workspace or the name of an item that has a subworkspace.

Attribute Name	Description
dxiov-status	The current status of the import object, read-only.
dxiov-import-object-busy	The activity state of the import object. This flag will be true while import is active or if an error occurred during an import. A menu choice exists to reset this flag if an error occurred during import.
dxiov-input-translation-workspace	The name of the workspace or name of the object superior to the workspace that contains translation objects.
dxiov-default-class-to-create	The default object to create if no translation between the OpenView description and G2 class can be determined.
dxiov-input-destination-workspace	The final destination of the top level objects. This may be the name of a workspace or the name of an object superior to the workspace where top objects will be transferred.
dxiov-new-class-destination-workspace	The name of a workspace or item superior to the workspace where new class definitions will be placed.
dxiov-file-retrieve-proc	The name of the procedure that requests the oobjprint from OpenView. This procedure and supporting procedures are transferred to the user module as part of the installation.
dxiov-file-retrieve-command	The command line string that will be issued to the local machine for requesting the oobjprint. This string is used by the dxiov-file-retrieve-proc.

Attribute Name	Description
dxiov-local-file-name	The directory and name of the destination of the ovobjprint file. This attribute will change depending upon the operating system type.
dxiov-default-column-height	The number of items placed in a column before starting the next column.
dxiov-vert-spacing	Space between columns of objects.
dxiov-horz-spacing	Space between rows of items.
dxiov-local-connected-classes	An array of class names that will be connected on workspaces, usually a list of nodes, routers, and IP devices. This must include the class name of the IP devices (card) for Reachability Analysis. Additional classes in this list define items that will be connected orthogonally on a workspace for graphic clarity. This list does not include classes of items that are connected diagonally as defined by OpenView.

dxiov-type-to-class-object

This object is used to translate a known text to a class definition. This is where you can specify which class gets instantiated based on a piece of text that gets parsed from the resulting ovobjprint file.

Attribute	Description
dxiov-type-string	A string to be matched in the resulting ovobjprint generated file
dxiov-type-class	The class to instantiate when the Dxiiov-type-string is matched within the file

Translations

Translation objects are located in the Navigator under System Models > Parsed MIBs. These are instances of the `dxiov-type-to-class-object` class. They are used to determine which class type to instantiate given a known text string.

There are two attributes in the `dxiov-type-to-class-object`:

Attribute	Description
<code>dxiov-type-string</code>	A text string containing a word or phrase that describes an object in the OpenView database.
<code>dxiov-type-class</code>	A symbol representing the name of class to instantiate when there is a match with the text defined in <code>dxiov-type-string</code>

Note IPRA contains instances of `dxiov-type-to-class-object`, which have already been configured.

Initializations

If using IPRA, initialization objects are located in the DXIOV Initializations workspace (XYZ-TOP-LEVEL > Application Objects > Initializations > Initializations > DXIOV Initializations).

xyz-dxiov-import: dxiov-file-retrieve-command

```
rsh OV-hostname -l username -n /usr/OV/bin/ovobjprint
```

Note where, the `rsh` may need to be changed to `remsh`, and the `ov-hostname` and the `username` should be already set during installation and testing (see page 379).

xyz-dxiov-import: dxiov-file-retrieve-name

```
/tmp/ovobjprint.txt
```

On UNIX, you may change it to:

```
/tmp/objprint.txt
```

On Windows:

```
C:\temp\objprint.txt
```

Initializations that could be used if merging OV Map Importer (DXIOV.KB) into your application include the following:

dxiov-mib-lookup

This translation names a procedure that can be used to lookup Enterprise to name translations.

File Transfer Routines

The requesting of information from OpenView is driven by a procedure, *xyz-retrieve-proc*, which spawns the command line function that requests the file.

In IPRA, this procedure can be found in the DXIOV Procedures workspace under the Reasoning Routines workspace.

If an *objprint.txt* file is to be manually transferred into the default directory, a dummy procedure (e.g., *xyz-retrieve-proc-dummy*) may be created that implies the file has been returned successfully. This procedure would be entered into the *dxiov-file-retrieve-proc* attribute of the *dxiov-import-object*.

Here is an example:

```
xyz-retrieve-proc-dummy (DXI: class dxiov-import-object, Command: text) =
(truth-value)
begin
  return TRUE;
end
```

Translation Objects

Translation objects are set up to define the class of item to create, depending upon the type of object stored by OpenView. To help you understand the structure of the translation objects, here is an example of an object defined by the *ovobjprint*:

OBJECT: 1024

FIELD ID	FIELD NAME	FIELD VALUE
10	Selection Name	"hou38"
11	IP Hostname	"hou38"
14	OVW Maps Exists	2
15	OVW Maps Managed	2
17	vendor	Sun(25)
27	isNode	TRUE
29	isComputer	TRUE
30	isConnector	FALSE
31	isBridge	FALSE
32	isRouter	FALSE
33	isHub	FALSE
36	isWorkstation	TRUE
51	isIP	TRUE

516	IP Status	Normal(2)
519	isIPRouter	FALSE
540	isSNMPSupported	TRUE
542	SNMP sysDescr	"SunOS houston38 5.5 Generic sun4m"
543	SNMP sysLocation	"Gensym-Conroe"
544	SNMP sysContact	"william"
545	SNMP sysObjectID	".1.3.6.1.4.1.11.2.3.10.1.2"
546	SNMPAgent	HP Solaris Sparc(20)
555	TopM Interface Count	1
561	TopM Interface List	"le0 Normal 195.93.147.38 255.255.255.0 0x080020736839 ethernet csmacd"
562	isMcClusterMember	FALSE
563	isCollectionStationNode	FALSE

When OV Map Importer reads the file, it looks for field names that begin with "is" and have a value of TRUE. These fields are then concatenated with hyphens (-). The translation object for the example has a type string of **Node-Computer-Workstation-IP-SNMPSupported**.

Once OV Map Importer finds a match, the type class of that translation is used as the base class of the object. The SNMP `sysObjectID` field defines the specific equipment. The initialization object for `DXIOV-MIB-LOOKUP` determines if there is a MIB lookup procedure to convert the `ObjectID` to a name. If the OXS module is not part of the knowledge base, the `sysObjectID` is appended to the end of the class string to determine the actual class of G2 object to create.

If OXS was loaded, OV Map Importer looks for an `oid-to-name-translation` that returns a name for this `ObjectID`, in this case, the returned name will be appended to the class string.

New Class Creation

New classes are created by OV Map Importer under two circumstances:

- No translation object could be found that matched the "is" fields in the file.
- The class of object to create does not exist.

When OV Map Importer does not have a translation object that applies to an object, it will create an object definition with a class name as determined from the above circumstances. The superior to that class will be the class named by the `default-class-to-create` attribute of the `import-object`. For the above case, with no translation object, the `object-definition` class name would be:

`Node-Computer-Workstation-IP-SNMPSupported_1.3.6.1.4.1.11.2.3.10.1.2`

If OV Map Importer has a translation object but cannot match the SNMP `sysObjectID` field, and an `oid-to-name` translation exists for that `ObjectID` then the class definition created would be:

`Node-Computer-Workstation-IP-SNMPSupported_Sun-SPARC(20)`

Otherwise, the name would be as in the first example.

The superior class of in this instance would be that named by the type class of the translation object.

The new class definitions created by OV Map Importer are used to allow the application to provide specific attributes and icons to different types of equipment.

OV Map Importer Operation

Building the Domain

To create the initial domain:

→ Display the Setup dialog and select the Domain Import tab.

A progress clock will be displayed during the import process. It will then disappear once the importation is complete.

To build the domain:

- 1 Create and transfer the `objprint.txt` file to the local machine.
- 2 Optionally, create and transfer the `layer-2-info.obj` file to the local machine.
- 3 Fill out the information in the HPOV Setup and Import section of the Domain Import tab of the Setup dialog. If you have exported the Layer 2 information from HPOV, be sure to select the checkbox to indicate this.

During the import process, you will notice devices being added to the Navitator under the System Models > Network Diagrams node.

Incremental Build

Receiving traps from OpenView whenever a new IP device is discovered supports incremental addition of objects to the domain. OV Map Importer will make between two and four calls to OpenView to determine the location and containment for the new device.

- The first call to OpenView requests information about the IP device (i.e. card), given the selection name that was received in the trap.
- The response will include the name of the superior node, the segment, and the subnetwork to which this device belongs.
- Additional requests will be made until a superior device is found to exist in the current domain.

Errors

If an error is encountered during import, the process will terminate. In this event, selecting the import will not offer the `dxiov-start-import` menu option but will instead offer the `dxiov-busy-reset` menu choice. Selecting `dxiov-busy-reset` will allow the import to be attempted again.

- The import object displays all setup errors caused by nonexistent workspaces or procedures before an import begins. Correct any setup errors and retry the import.
- During the import, objects are created and transferred to the subworkspace of the import object along with an `object-receiver`. The receiver is created to maintain information read from the file while import is in progress.
- If an error occurs during import, the attribute display shows the last working state of the import at any given time. Most errors occur during reading of the file and can quickly be resolved by looking at the last object created by OV Map Importer (the bottom right object on the subworkspace of the import object), and finding the next object in the `objprint.txt` file. Phantom lines in the `objprint.txt` file are not uncommon.
- An import error will leave the `dxiov-import-object-busy` flag true. Selecting the import object again will provide a menu choice to reset this flag and will retry import upon the next selection.

Notes on GDXI

- The attribute `dxiov-default-class-to-create` may be set to `ipra-domain-object` to build everything that is referenced in the file. The objects of this type are unrecognized object in the `objprint.txt` file. To inhibit building them, the attribute may be set to `dxiov-excluded-object`, these will not show up in the final map.
- Translations for specific items not to be created reference the class `dxiov-excluded-object`.
- If an object has an OID, OV Map Importer will look up an OID- to- name translation and then see if a class of that name exists. If so, it will build one. If the full OID does not map to a class, the OID string is shortened by one dot (`.`), starting at the far right, and tried again. This goes on until a class is found or the string is back to `.1.3.6.1.4.1`.
- The `connected-local-classes` attribute has been removed. Determination of connected classes is now done through multiple-inheritance by mixing in the class `dxiov-local-connected-object`. All items that are a `dxiov-local-connected-object` and reside on the same workspace will be connected together.

Ping Manager

Introduces the Ping Manager and describes how to install, setup, and run the Ping Manager.

Introduction **389**

Components **390**

Running the Ping Manager **390**

The Remote Procedure Calls **391**

Application Development **394**

Sample Procedures and Actions for pingmgr.kb **394**



Introduction

If internetworks were flawless, datagrams would always be routed to their intended destination with no errors, excessive delays, or retransmissions. Unfortunately, this is not the case. Internet Protocol (IP) provides a connectionless service to the attached hosts, but requires an additional module, known as the Internet Control Message Protocol (ICMP) to report any errors that may occur in the processing of those datagrams. The protocol is also used to test the path to a distant host, which is known as “pinging”.

IP datagrams contain ICMP messages. The ICMP messages contain valuable information about the network status. The Ping Manager uses the Echo message (ICMP Type = 8), which tests the communication path from a sender to receiver via the Internet. The sender transmits an Echo message, which may contain an identifier and a sequence number as well as data. When the intended destination

receives the message, it reverses the source and destination addresses, recomputes the checksum, and returns an echo reply (ICMP Type = 0). The contents of the data field (if any) would also return to the sender.

The OpEx Ping Manager issues ICMP Echo Pings to network devices according to specifications provided through a G2-based graphical user interface. The configuration defines attributes such as `ping-frequency`, `time-out-interval`, and `maximum-retries` allowed. The Ping Manager maintains scheduled, multi-threaded pinging for the devices and only reports to G2 in the event of a status change. The G2 application can also perform on-demand pings through the Ping Manager.

The Ping Manager is provided as a stand-alone executable code, interfaced to G2 using the standard "gateway" (GSI) approach. A support KB is also provided.

The stand-alone executable code is supported under HP-UX 10.x, Sun Solaris 2.5, SunOS 5.5, and IBM AIX 4.x. The KB portion runs on any platform supporting G2 5.1 Rev. 1e, such as Windows, HP-UX, Solaris, AIX.

Components

The Ping Manager consists of two components, the Ping Manager Executable and the Ping Manager G2 Knowledge Base.

The Ping Manager executables are located in the dependent directory on the distribution CD. For each platform supported there will be a directory containing the executable (`pingmgr` or `PingMgr.exe` for Windows) and the knowledge base (`pingmgr.kb`).

To install the Ping Manager executable:

- ➔ Use the installation CD and make sure the Ping Manager component is selected.

Running the Ping Manager

To run the Ping Manager executable on Unix:

```
$ ./pingmgr {port-number} {-t[imeout] n} {-r[etries] m}
```

To run the Ping Manager executable on Windows:

```
run PingManager.exe {port-number} {-v[erbose]} {-t[imeout] n}  
{-r[etries] m}
```

The `{port-number}` is optional. If the port-number is not specified, then the default port-number 22054 will be used for UNIX and port 2500 will be used for Windows. The `-t` and `-r` are also optional. These options allow users to set their desired timeout and maximum retries. Where `n` and `m` are any integer. If these

options are not set by the user, the default timeout is 12 and the maximum retries is 1.

Note Only perform these steps if you are not using the IPRA product and you want to use the Ping Manager in an existing application.

To load and run the pingmgr KB:

- 1 Pause G2.
- 2 Merge *pingmgr.kb* into your application.
When running Integrity, this module is included in the module heirarchy.
- 3 A `gsi-interface` object, `ping-manager-interface`, exists in *pingmgr.kb*. Change the `gsi-connection` to read:

```
tcp-ip host "host-name-or-ip-address" port-number port-number
```

where:

- *host-name-or-ip-address* is the hostname or IP address of the machine running the Ping Manager executable.
- *port-number* is the port number the Ping Manager executable is listening on. Use it to connect to the Ping Manager executable process.

Note To establish a connection, the user needs to re-configure the `ping-manager-interface` if either the *host-name-or-ip-address* is changed or the *port-number* is changed, or after the Ping Manager is restarted, even when there are no changes to the configuration.

A `gsi-interface` status indicator indicates the following connection status:

- 2 Not connected.
- 1 Connection is in progress.
- 2 Connected.

The Remote Procedure Calls

The Ping Manager provides the following RPCs (remote procedure calls).

Setting the Device Configuration for a Ping Manager

Use the following RPCs to configure a device for the Ping Manager.

pm-add-device-config-rpc

*(ext-name: text, ip-address: text, interval: integer {seconds},
time-out: integer {seconds}, retries: integer)*

Adds a device to the Ping Manager's device list.

pm-delete-device-config-rpc

(ext-name: text)

Deletes a device from the Ping Manager's device list.

pm-get-device-config-rpc

*(ext-name: text) => (dev-name: text, ip-address: text,
interval: integer {seconds}, time-out: integer {seconds},
max-retries: integer, status: text)*

Get a device's configuration from the Ping Manager's device list.

pm-load-config-file-rpc

(filename: text)

-> devices-loaded: integer {number of devices loaded}

Loads a configuration file that contains device(s) configuration information into a Ping Manager.

Note See "A Sample Configuration File"

pm-write-config-file-rpc

(filename: text)

-> devices-saved: integer {number of devices written}

Writes the Ping Manager's current device(s) configuration information to a file. The file is saved in the same location as the Ping Manager executable.

Changing a Device's Configuration for the Ping Manager

Use the following RPCs to configure a device for the Ping Manager.

pm-change-ip-address-rpc

(ExtName: text, NewIpAddress: text)

Changes the ip-address.

pm-change-poll-time-out-rpc

(ExtName: text, PollTimeOut: integer)

Changes the poll-time-out.

pm-change-polling-info-rpc

(*ExtName*: text, *NewInterval*: integer {seconds},
NewTimeOut: integer {seconds}, *NewRetries*: integer)

Changes the poll-interval, poll-time-out, and number-of-retries.

pm-change-poll-interval-rpc

(*ExtName*: text, *NewPollInterval*: integer {seconds})

Changes the poll-interval.

pm-change-max-retries-rpc

(*ExtName*: text, *NewRetries*: integer)

Changes the number-of-retries.

pm-manage-device-rpc

(*ExtName*: text)

Initiates periodic polling of a device.

pm-unmanage-device-rpc

(*ExtName*: text)

Cancels periodic polling of a device.

pm-send-ping-request-rpc

(*ExtName*: text, *Status*: integer)

Sends a ping request to a device. To start polling without waiting for a state change, specify *option* as 1. To start polling only when there is a state change, specify *options* as 0.

pm-do-demand-poll-rpc

(*ExtName*: text, *Address*: text)

Starts demand polling.

pm-do-device-poll

(*gsi-interface*: text, *device-name*: text, *ip-address*: text)
 -> status: text

Starts demand polling of a device and returns the status, which is one of: "OK", "FAIL", "CAN-NOT-PING", "UNKNOWN"

pm-dump-agent-config-rpc

()

Displays the configuration information for all devices in the device list of the Ping Manager.

pm-kill-agent-rpc

()

Stops the Ping Manager from G2.

Application Development

The Ping Manager supports both demand polling and exception-based network management architectures.

Demand Polling

The following example polls a device on demand:

```
status-text: text;
status-text = pm-do-device-poll("my-gsi-interface", "my-device-name",
"1.1.1.1");
```

Periodic Polling

Follow a call to `pm-add-device-config-rpc` with one of the following calls, depending on whether you want to poll with or without a state change, respectively:

```
call pm-send-ping-request-rpc ("my-device-name", 1)
call pm-send-ping-request-rpc ("my-device-name", 0)
```

Sample Procedures and Actions for pingmgr.kb

To add a device:

→ Start `pm-add-device-config-rpc("mypc.com", "123.45.6.78", 240, 10, 3)` across `ping-manager-interface`.

To load a configuration file:

→ Start `pm-load-device-config-rpc("/path/ping.cfg")` across `ping-manager-interface`.

where, the path on UNIX, may be:

`/tmp` or `~myhome`

on Windows, may be:

`C:\data\`

A Sample Configuration File

The configuration file contains polling information on devices. Each device is defined on a separate line containing five fields:

- device name

- ip-address
- interval
- time-out
- maximum retries

If the device is not in the configuration file, the default number of retries and default time-out interval are used.

The Ping Manager interprets a blank space as a delimiter between fields on a single line. You should surround values containing spaces with double quotation marks ("").

Blank lines are allowed to separate device configurations.

Comments are designated with a '#' character and can be used at the beginning of a line or at the end of a configuration for a device. Any text following a '#' on a single line is ignored.

Example

```
# File name: ping.cfg
#
# The configuration fields are described as follows:
# "device name" ip-address interval time-out max-retries

# A pair of quotes must be used if a device name has any blank space(s) in
it.
"My new computer: enterprise" 123.45.6.78 120 10 3
# device name without blank spaces, quotes are optional
abc.gensym.com 123.45.6.79 240 12 2
"def.gensym.com" 123.45.6.80 180 20 3
# blank lines are allowed in the configuration files

xyz.gensym.com 123.45.6.99 120 10 1
# end of ping.cfg
```

Example of a Procedure to a get configuration status

```
get-device-status(ext-name: text)
dev-name, ip-address, status: text;
interval, time-out, max-retries: integer;
begin
```

```
dev-name, ip-address, interval, time-out, max-retries, status =  
    call pm-get-device-config-rpc(ext-name) across ping-manager-interface;
```

```
inform the operator that "[dev-name]: [ip-address]:[interval]:  
[time-out]:[max-retries]:[status]";
```

```
end
```

Example of an Action-Button to Invoke get-device-status

```
start get-device-status("my-device-name.gensym.com");
```


G2-SNMP Bridges

Chapter 20: Overview of the G2-SNMP Bridges

Describes the G2-SNMP Bridges and their intended applications and provides an overview of the Integrity product family.

Chapter 21: Installation and Startup

Describes how to install, authorize, and execute the G2-SNMP Bridges software.

Chapter 22: G2-SNMP Bridge Setup

Describes how to configure the G2-SNMP Bridge, to set up Integrity to process SNMP traps, and to perform additional processing in response to incoming traps.

Chapter 23: G2-SNMP Bridges API

Provides a listing of the G2-SNMP Bridges APIs, remote procedure calls, procedures, and functions.

Chapter 24: Reporting Errors

Describes how to report bugs in a G2-SNMP Bridge to Gensym customer support.

Overview of the G2-SNMP Bridges

Describes the G2-SNMP Bridges and their intended applications and provides an overview of the Integrity product family.

Introduction **399**

Applications **401**

Features and Benefits **401**

Acquiring Data **401**

Building a G2-SNMP Bridge Application **402**

G2-SNMP Bridges and the Integrity Product Family **403**

Enhancements **404**



Introduction

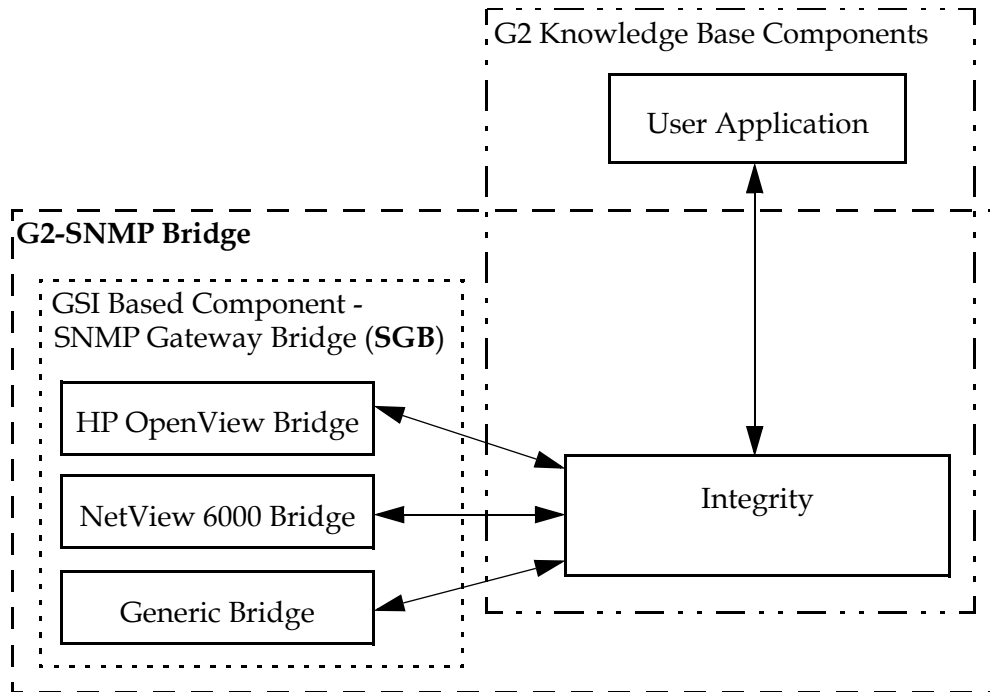
The G2 Simple Network Management Protocol (G2-SNMP) Bridges enable a user application to communicate with devices that support SNMP, through the HP OpenView Network Management System, the NetView 6000 Network Management System, or directly through Gensym's G2-SNMP Generic Bridge. This guide describes the functionality of the G2-SNMP Bridges and explains how to use them in a G2 application.

The G2-SNMP Bridges provide a set of functions to perform SNMP transactions (SET, GET, and GET NEXT) and to send and receive SNMP traps.

The G2-SNMP Bridges use the **G2 Standard Interface (GSI)**. GSI is a toolkit for creating bridges between G2 and external systems. Gensym has used GSI to create

the G2-SNMP Bridges for you. See the *G2 Standard Interface User? Guide, Version 3.2* for additional information about GSI.

The G2-SNMP Bridges are made up of three individual GSI based bridges which are the HP OpenView, NetView 6000, and Generic Bridge. Each of these bridges interface to G2 using an Integrity application. The user's application will in turn interface with the G2-SNMP Bridges through the Integrity application. The following figure shows the components of the G2-SNMP Bridges.



Note Unless otherwise specified, the term **SGB** (SNMP Gateway Bridge) refers to the GSI based component of the G2-SNMP Bridges and the term Integrity refers to the G2 knowledge base component of the G2-SNMP Bridges.

Applications

The G2-SNMP Bridge provides a general infrastructure for the development of applications for Simple Network Management Protocol (SNMP) based network management activities.

Features and Benefits

The G2-SNMP Bridge toolkit provides an End User, Value Added Reseller (VAR) or an Integrated System Vendor (ISV) with an environment that will:

- Reduce application development time
- Promote development of reusable objects
- Allow deployment of the application across diverse hardware platforms

Applications developed with the G2-SNMP Bridge will be:

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

Acquiring Data

The SGB can send information to Integrity in either of two ways

- **Mode 1:** The SGB receives traps from the HP OpenView SNMP trap daemon (*trapd*) or the Generic Bridge trap receiver and forwards them to Integrity by making remote procedure calls to Integrity **receiver procedures**. Calls to the receiver functions are **non-blocking** -- that is, both the SGB and Integrity can make other calls while the call to the Integrity receiver procedure is being processed.
- **Mode 2:** Integrity performs SNMP requests and sends traps through the SGB by making remote procedure calls (RPC's) to procedures in the SGB. The SGB returns data to Integrity only when Integrity solicits the data. Calls to remote procedures in the SGB may be **blocking** -- that is, neither the SGB nor Integrity can make other procedure calls while the remote procedure call is being processed -- or non-blocking.

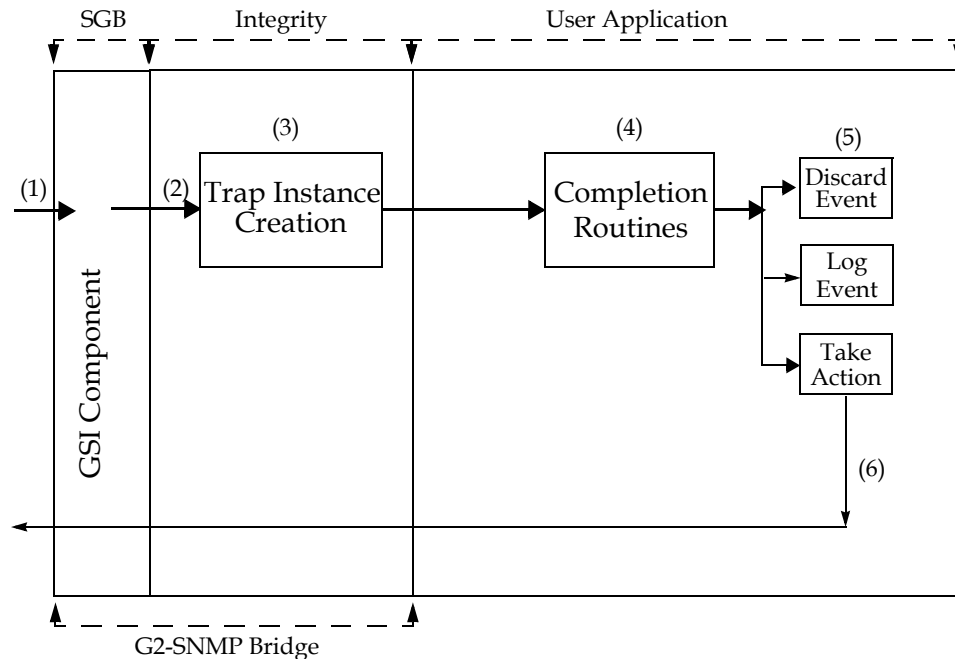
A typical G2-SNMP Bridge application requires that two SGB processes be running, one to send traps to Integrity receiver procedures, and one to perform SNMP transactions at the request of Integrity.

If you are running the SGB to receive traps (Mode 1), the HP OpenView trap daemon (*trapd*) or Generic Bridge trap receiver must be running on the same machine as the SGB.

Building a G2-SNMP Bridge Application

The following diagram is a block representation of the architecture of an application using a G2-SNMP Bridge. The G2-SNMP Bridge is capable of being fully integrated with Integrity.

In the following diagram the arrows show the progress of an event through a typical user application.



The event moves through an application as follows:

- 1 An event enters the system through the SGB.
- 2 The event is passed to Integrity. Here further parsing and decoding is done as needed.
- 3 Integrity creates the appropriate trap class instance.
- 4 Integrity calls the appropriate user completion routine.
- 5 The completion routine sometimes completes the parsing and decoding of the event. In the completion routine you can discard the event, or perform further processing of the event such as logging the event. The completion routine is a G2 procedure written by you.
- 6 You can send an SNMP trap to an external system as part of the application.

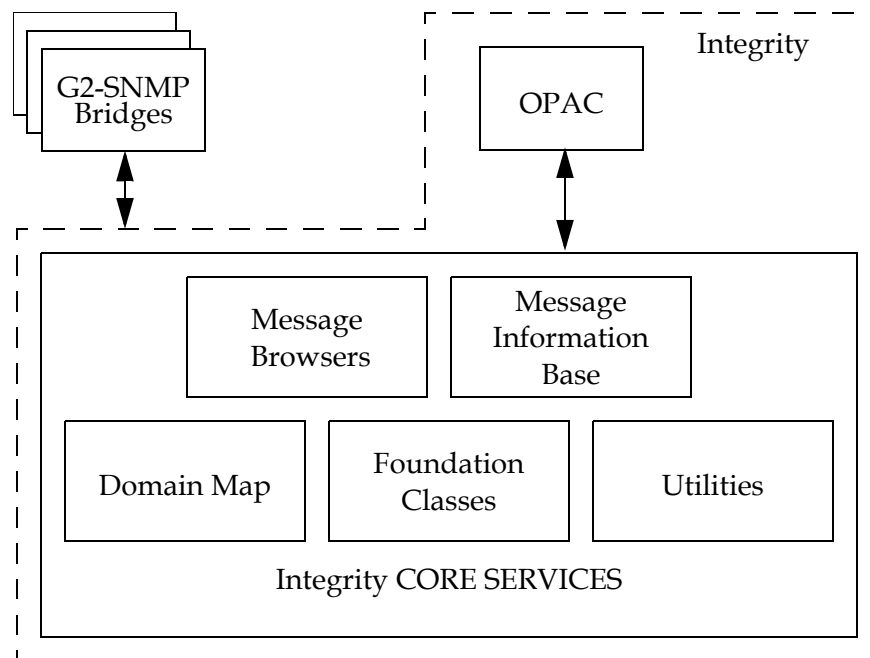
G2-SNMP Bridges and the Integrity Product Family

The G2-SNMP Bridges are a member of the Integrity (formerly Operations Expert (OpEx)) product family. The Integrity product family comprises Network, System, Service, and Application Management tools and applications.

The Integrity product family presently includes the following:

- Integrity - A tool for building alarm management applications.
 - CORE Services
 - Contains the Integrity foundation classes
 - A message management system; servers and browsers
 - Tools for building a representation of your managed objects
 - A set of development utilities.
 - Network analysis utilities
 - Logging utilities
- OPAC - A graphical programming language.
- G2-SNMP Bridges - An SNMP interface for the Integrity product family.

The diagram below shows the G2-SNMP Bridges and their relation to the components of the Integrity product:



Integrity is a product designed to help companies monitor and control their operations to increase the availability and service levels of their distributed, mission-critical environments. Integrity currently includes OPERator ACTIONS (OPAC), Integrity Core Services (OPCS), and several demonstrations, and will include other components in the future.

In addition, 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.

For additional information on the Integrity product family contact your Gensym sales representative.

Enhancements

Enhancements in the G2-SNMP bridge (starting with version 3.1) include the following:

- Support for filtering of traps from specified hosts

This version of the G2-SNMP bridge supports filtering of incoming traps based on the agent IP address bound with the trap. You may specify either the agent hostname or agent IP address for filtering purposes. If the agent hostname is specified, it must be possible to map it to an IP address through the *gethostbyname()* Unix library function.

This additional filtering capability is in addition to the existing filtering capability based on enterprise, generic, and specific ID's for the trap.

- Support for specifying agent hostnames and agent IP addresses using regular expression matching

This version of the G2-SNMP bridge supports specifying agent hostnames and agent IP addresses using *glob*-style regular expression matching.

- Support for adding new agent hostnames/IP addresses using G2 RPCs and toggling filtering status of individual agents

This version of the G2-SNMP bridge allows you to add agent hostnames and/or agent IP addresses (including those specified as regular expressions) to the filtered hosts list using a G2-based RPC. A complementary RPC allows you to toggle off the filtering status of any entry in the agent filter list.

- Support for specifying host filtering mode on the command line

This version of the G2-SNMP bridge supports specifying the agent filtering mode, 0 (OFF), 1 (PASS), or 2 (BLOCK) on the command line using the '*-f <mode>*' command line option.

- Support for specification of trapd.conf preprocessed file (ppd) names through the bridge command line

This version of the G2-SNMP bridge supports two methods of specifying the *trapd.conf* preprocessed file (ppd) name: through the existing `G2_PPD_FILENAME` environment variable and through a new bridge command line option, `-p <ppd filename>`. When the `-p` option is specified, it overrides the `G2_PPD_FILENAME` environment variable, if it is defined.

- Support for passing the SNMP community name string through the bridge to G2

This version of the G2-SNMP bridge now passes the community string (and in the non-NT versions of the bridge, the length of the community string) from received traps through to G2.

- Support for specifying the path and file name for the filtering log file

This version of the G2-SNMP bridge supports specifying the path and file name for the filtering log file through a new bridge command line option, `-l <path and log file name>`.

- Bridge returns 'General failure on agent' code

This release of the G2-SNMP bridge returns the correct error code associated with the 'General failure on agent' error.

- Bridge passes variable values for variable bindings in which the variable type is 'object identifier'

This release of the G2-SNMP HP OpenView bridge under HP-UX passes the correct variable values for variable bindings in which the variable type is 'object identifier'. The bug fix is restricted to the GSI-based portion of the bridge. It does not affect the Integrity support KB. Previously, the bridge passed either a null string (") or a random value from memory (such as an IP address) in place of the proper value of the variable. The bridge now correctly passes varbind variable values of type 'object identifier'.

Installation and Startup

Describes how to install, authorize, and execute the G2-SNMP Bridges software.

Introduction **407**

UNIX Platform Installation **407**

Authorizing the G2-SNMP Bridges **415**

Executing the G2-SNMP Bridge **416**

Connecting G2 to the GSI Bridge Process **419**



Introduction

This chapter describes how to install, start, and use the G2-SNMP Bridge.

UNIX Platform Installation

You can install the G2-SNMP Bridges software from tape or from CD-ROM.

Installing from Tape

The distribution tape contains the necessary files for the G2-SNMP Bridges software.

The steps are:

- 1 Establish an installation directory.
- 2 Insert the tape into the tape drive.

- 3 Extract the **tar** archive.
- 4 Remove the tape and store it in a safe place.

The G2-SNMP Bridges software is a multiple-file **tar** archive. It contains two archives in the following order: SNMP Gateway Bridges (GSI-based component) and Integrity.

Install the archives, using the **tar** facility to read the **tar** archives. Use the **mt** (magnetic tape control) and **fsf** (forward space over current end of file marks) commands to move through the archives.

Note The following instructions use a percent and pound sign (**%** and **#**) to designate the system prompt. When entering a command, type what follows the sign; do not type the sign itself. The percent sign designates the user prompt and the pound sign designates the super user prompt.

Determining the Device Name

UNIX commands that manipulate tapes do so by specifying an option whose value is a **device name**. This name indicates both the tape drive to use and the mode in which to use it. The option letter and the device name differ on different platforms. The following table lists the options and device names for installing the G2-SNMP Bridges software on various platforms:

UNIX Options and Device Names

Workstation	option	device-name
HP 9000 Series 300, 400, 700, 800	<i>-t</i>	<i>/dev/rmt/0mn</i>
IBM RISC System/6000	<i>-f</i>	<i>/dev/rmt0.1</i>
Sun SPARCstation	<i>-f</i>	<i>/dev/nrst0</i>
Sun Solaris	<i>-f</i>	<i>/dev/rmt/0n</i>

You can install the SNMP Gateway Bridge software either in an existing directory or in a new directory. We recommend that you install the SNMP Gateway Bridge software under a new subdirectory, for example, *g2snmp-23r0*. If you prefer to use an existing directory, be sure to make appropriate backups because some of the existing files may be overwritten.

To install the SNMP Gateway Bridge software from tape:

- 1 Login as the user who should own the installed files.
- 2 Create a writable directory where you want to install the SNMP Gateway Bridge software.

For example, if you want to install the SNMP Gateway Bridge software in a new subdirectory *g2snmp-23r0* under */usr/gensym*, enter the following commands:

```
% cd /usr/gensym
% mkdir g2snmp-23r0
% cd g2snmp-23r0
```

If you want to install the SNMP Gateway Bridge software in an existing directory, for example, */usr/gensym/g2snmp*, enter:

```
% cd /usr/gensym/g2snmp
```

- 3 Insert the tape into the drive.
- 4 If necessary, rewind the tape by executing:


```
% mt option no-rewind-device-name rewind
```
- 5 To install the SNMP Gateway Bridge software in this directory, execute the following command:

```
% tar -xvf no-rewind-device-name
```

Your installation of the SNMP Gateway Bridges software is complete, continue with the installation of the Integrity software.

You can install the Integrity software either in an existing directory or in a new directory. We recommend that you install the Integrity software under a new subdirectory, for example, *oxs-23r0*. If you prefer to use an existing directory, be sure to make appropriate backups because some of the existing files may be overwritten.

To install the Integrity software from tape, see the *readme-g2.html* file.

If you are installing the G2-SNMP “Generic” Bridge, continue with the following steps to complete your installation of the G2-SNMP Bridges software.

G2-SNMP “Generic” Bridge Additional Installation Steps

The G2-SNMP “Generic” Bridge requires the following additional steps to complete the installation.

Note You must log in as *root* user to complete the installation of the G2-SNMP “Generic” Bridge.

To become a root user:

- 1 At the UNIX prompt, enter:

```
% exit
```

This logs you out of your current userid.

- 2 At the login prompt, log in as *root*.

For example:

```
login: root
```

- 3 Enter the password for *root*.

Entering a correct password will grant you *root* user privileges.

Note The components of the G2-SNMP “Generic” Bridge are bundled in a file called *pkg_tar.Z*. An installation program, *install.brg*, which unbundles these G2-SNMP “Generic” Bridge components, is included on the distribution tape.

The file *pkg_tar.Z* may have inadvertently been renamed to *pkg_tar.z*. If so, rename the file with an upper case Z, *pkg_tar.Z*.

- 4 Using the UNIX command *cd*, change to the directory where the G2-SNMP “Generic” Bridge files have been installed from the distribution tape, for example:

```
# cd /usr/gensym/g2snmp-23r0/
```

- 5 At the UNIX prompt, execute the command:

```
# ./install.brg
```

This command uncompresses the *pkg_tar.Z* file and extracts the following files:

- The G2-SNMP “Generic” Bridge executable, “*g2snmpgn.*”
- The G2-SNMP “Generic” Bridge trap receiver, */usr/local/bin/straps*
- An auxiliary system file, */usr/local/bin/ntping* and
- Other files common to all SNMP Gateway Bridges

- 6 Log out as *root*, enter:

```
# exit
```

Your installation of the G2-SNMP “Generic” Bridge software is complete.

Installing from CD-ROM

You can also install the G2-SNMP Bridges software from a CD-ROM disk.

The essential steps are:

- 1 Insert **the disk** into the CD drive.
- 2 Mount the CD (except under Solaris, which automounts).
- 3 Run the Install script, this script will guide the user through the installation process, and install the product(s) that you need.
- 4 Quit the Install script.
- 5 Unmount the CD.

Note The following instructions use a percent and pound sign (% and #) to designate the system prompt. When entering a command, type what follows the sign; do not type the sign itself. The percent sign designates the user prompt and the pound sign designates the super user prompt.

To mount the CD:

- 1 Insert the CD into the CD drive.
- 2 Under Solaris, and on a Silicon Graphics IRIS-4D machine with an SGI drive, the CD automounts; no further action is required. On other platforms, you must login as root and mount the CD. First execute:

```
% su root
```

The system prompts you for your system password.

- 3 Enter your password.

A correct password grants you *root* user privileges.

- 4 Execute the *mount* command shown for your platform in the following table:

UNIX Operating System or Workstation Platform	Example Mount Command
HP 9000 Series 700, 800	<code>mount -rt cdfs /dev/dsk/cdrom /cdrom</code>
IBM RISC System/6000	<code>mount -rt cdrfs /dev/cd0 /cdrom</code>
Sun Solaris	<code>mount -rF hsfs /dev/sr0 /cdrom</code>

- 5 Log out as *root*.

To install the G2-SNMP Bridges software from a CD-ROM disk:

- 1 Log in as the user who should own the installed files.

Caution Do *not* install the G2-SNMP Bridges software as *root* unless you want your installed files to be owned by *root*.

- 2 Locate the G2-SNMP Bridges Install script in the *root* directory of the disk. Under Solaris, enter:

```
% ls /cdrom/cdrom0
```

On any other platform enter:

```
% ls /cdrom
```

The general name of the Install script is *unixinst*. Depending on your system, the script will appear as one of the following:

```
unixinst
UNIXINST
UNIXINST.;1
unixinst.;1
unixinst.-1
```

Note Installation script filenames can appear in one of several ways due to the varying levels of support by CD-ROM drivers and operating systems for the ISO 9660 and Rock Ridge formats.

- 3 Execute the Install script as it appears on your system. Give a fully qualified pathname beginning with */cdrom*, and surround the name of the script with double quotes if it contains any trailing nonalphanumeric characters. For example:

```
# /cdrom/"unixinst.;1"
```

Note Under Solaris, the pathname would begin with */cdrom/cdrom0* rather than */cdrom*.

- 4 The Installer prompts you for the name of the mount drive. Under Solaris, enter:

```
/cdrom/cdrom0
```

On any other platform, enter:

```
/cdrom
```


To install the SNMP Gateway Bridges software from a CD-ROM disk

- ➔ The Installer prompts you with a list of products to install. Install products one at a time by selecting each desired product.
 - For each product, the Installer proposes a directory in which to install the product. You should accept the default location unless you have a specific reason not to.

Note The destination directory *must* be a new directory. It is recommended that you install the software in `/usr/gensym/g2snmp/`. If a directory with this name already exists, move its contents to another directory (for example, `/usr/gensym/g2snmp.bak/`), leaving the `/usr/gensym/g2snmp/` directory empty.

- If you prefer to install in a non-default directory, you can type in its name. The directory must *not* already exist; the Install script will create it.
- Installing some software products prompts the Installer to offer to install other, supporting products. You should accept such installation unless you have a specific reason not to.

Your installation of the SNMP Gateway Bridges software is complete, continue with the installation of the Integrity software.

To install the Integrity software from a CD-ROM, see the `readme-g2.html`.

To unmount the CD drive:

- 1 At the UNIX prompt, enter:

```
# su root
```

The system prompts you for your system password.

- 2 Enter your password.

A correct password grants you `root` user privileges.

- 3 At the prompt, unmount the CD drive by executing:

```
# umount /cdrom
```

- 4 Log out as `root`.

- 5 Store the CD safely away from heat, smoke, dust, and anything that might scratch its surface.

Your installation of the G2-SNMP Bridges software is complete.

If you are installing the G2-SNMP “Generic” Bridge continue with the following steps to complete the installation of the G2-SNMP Bridges software.

G2-SNMP “Generic” Bridge Additional Installation Steps

The G2-SNMP “Generic” Bridge requires the following additional steps to complete installation.

Note You must log in as *root* user to complete the installation of the G2-SNMP “Generic” Bridge.

To become a root user:

- 1 At the UNIX prompt, enter:

```
% exit
```

This logs you out of your current userid.

- 2 At the login prompt, log in as *root*.

For example:

```
login: root
```

- 3 Enter the password for *root*.

Entering a correct password will grant you *root* user privileges.

Note The components of the G2-SNMP “Generic” Bridge are bundled in a file called *pkg_tar.Z*. An installation program, *install.brg*, which unbundles these G2-SNMP “Generic” Bridge components, is included on the distribution tape. The file *pkg_tar.Z* may have inadvertently been renamed to *pkg_tar.z*. If so, rename the file with an upper case *Z*, *pkg_tar.Z*.

To complete the installation of the G2-SNMP “Generic” Bridge:

- 1 Using the UNIX command *cd*, change to the directory where the G2-SNMP “Generic” Bridge files have been installed from the distribution tape.

- 2 At the UNIX prompt, execute the command:

```
# ./install.brg
```

This command uncompresses the *pkg_tar.Z* file and extracts the following files:

- The G2-SNMP “Generic” Bridge executable, “*g2snmpgn.*”
- The G2-SNMP “Generic” Bridge trap receiver, */usr/local/bin/straps*
- An auxiliary system file, */usr/local/bin/ntping* and
- Other files common to all SNMP Gateway Bridges

- 3 Log out as *root*.

Your installation of the G2-SNMP “Generic” Bridge software is complete.

Authorizing the G2-SNMP Bridges

You must authorize your G2-SNMP Bridge before you execute it. To obtain authorization codes, call Gensym's Production and Licensing group in Cambridge, MA (or contact your account representative).

Before you call, be sure to have the hostname (available through the UNIX *hostname* command) and machine id (usually the CPU serial number, available through the UNIX *uname -a* or *hostid* command) of your workstation available.

Authorizing the SNMP Gateway Bridge

Production and Licensing will generate authorization codes that you must enter into your *gsi.ok* file, located in the same directory as the SNMP Gateway Bridges executable. A sample *gsi.ok* file, *samp_gsi.ok*, is provided with the SNMP Gateway Bridges.

To create a valid *gsi.ok* file

- 1 Using the UNIX *cp* command, make a copy of the *samp_gsi.ok* file called *gsi.ok*, enter:

```
% cp samp_gsi.ok gsi.ok
```

- 2 Modify the *gsi.ok* file using a text editor such as *vi*. The *gsi.ok* file is formatted as:

```
<hostname> <machine id> <bridge name> <bridge version> <codes>
```

where:

- *<hostname>* is the hostname of your workstation;
- *<machine id>* is the machine id obtained through the UNIX *uname -a* or *hostid* command;
- *<bridge name>* is either *G2-HPOV-SNMP* for the G2-SNMP HP OpenView Bridge or *G2-GNRC-SNMP* for the G2-SNMP Generic Bridge;
- *<bridge version>* is the version of the bridge; and
- *<codes>* are a set of integer codes obtained from Gensym Production and Licensing.

For example, a G2-SNMP HP OpenView Bridge Version 2.3 Rev. 0 *gsi.ok* file might look like:

```
hou51 2011005381 G2-HPOV-SNMP V2 5476642 3470987 234965 2219873
```

while a G2-SNMP Generic Bridge Version 2.3 Rev. 0 *gsi.ok* file might appear as:

```
hou51 2011005381 G2-GNRC-SNMP V2 2376642 3980987 643965 9829873
```

- 3 Save the *gsi.ok* file to disk. It is recommended that the file be saved in the same directory as the SNMP Gateway Bridges executable files.

Note The actual values of the entries in the *gsi.ok* file may vary, but the format will remain the same.

When the *gsi.ok* file is in place, you must set the *GSI_ROOT* UNIX environment variable to the directory in which the *gsi.ok* file is located. For example, at the UNIX prompt, type:

```
% GSI_ROOT=/usr/gensym/g2snmp
% export GSI_ROOT
```

Note The exact commands needed to set UNIX environment variables vary widely among different workstations and different versions of the UNIX operating system. Please consult the documentation for your workstation's operating system to determine the proper commands needed to set the *GSI_ROOT* environment variable.

You can now execute your SNMP Gateway Bridge.

Executing the G2-SNMP Bridge

Executing the SNMP Gateway Bridge

Invoke the SNMP Gateway Bridge with the following command:

```
g2snmpov. <HP-OV version> <port-number>
```

or

```
g2snmpgn. <port-number>
```

where:

HP-OV Version

```
g2snmpov. 400
```

The SNMP Gateway Bridge executable for HP OpenView v4.00 on the Sun SPARC/Solaris platform.

<i>g2snmpov.401</i>	The SNMP Gateway Bridge executable for HP OpenView v4.01 on the HP 9000/HP-UX platform.
<i>g2snmpov.410</i>	The SNMP Gateway Bridge executable for HP OpenView v4.10 on the HP 9000/HP-UX platform.
<i>g2snmpov.4</i>	The SNMP Gateway Bridge executable for NetView 6000 on the IBM RS/6000/AIX platform.
<i>g2snmpgn.</i>	The SNMP Gateway Bridge executable for the G2-SNMP Generic Bridge on the HP9000/HP-UX platform

Note: The period (.) at the end of '*g2snmpgn.*' is required.

<i>port-number</i>	The TCP/IP socket number over which G2 and the SNMP Gateway Bridge communicate. The get/set SNMP Gateway Bridge is usually started with a port number of <i>22041</i> . The trap receive SNMP Gateway Bridge is usually started with a port number of <i>22044</i> .
--------------------	--

For example:

```
% g2snmpov.400 22041
% g2snmpov.401 22044
% g2snmpov.410 22064
% g2snmpgn. 22051
```

Messages written to the terminal window describe the status of the SNMP Gateway Bridge during start-up and execution.

Finding an Available Port

If the port that you specify in the start-up command line (for example, '*22041*') is in use or has not yet been released when you restart the SNMP Gateway Bridge, you see an error message of the following form:

```
"TCP/IP: Unable to find listener on port 22041"
```

If the port has not been released, you may need to wait for as long as 120 seconds or more for the system to release it. If the port is in use and you do not want to wait for the port to be freed, start the SNMP Gateway Bridge using a higher port

number (e.g. 22051 or 22054). You will likely need to change the port number in your G2 application as well.

Running SNMP Gateway Bridges as Background Processes

The SNMP Gateway Bridge writes status messages to the terminal window in which it is started. If you start the SNMP Gateway Bridges as background processes (by appending an ampersand [&] to the end of each start-up command line), it is recommended that you start each process (get/set and trap receive) in a separate terminal window. This helps to avoid confusion in interpreting any status messages produced by the SNMP Gateway Bridges.

Executing the Integrity Application

You must be running G2 to run Integrity successfully.

To load Integrity:

- 1 Launch G2, using the usual method of starting G2 for your system.
For more information on loading and starting a knowledge base, see the *G2 Reference Manual*.
- 2 Load your Integrity application, which must include GSNMP and GTRAP.

To see a demonstration of Integrity's SNMP capabilities:

➔ Load the file called `oxs_demo.kb` from the Integrity *examples* directory.

Once the KB has been loaded, if the workspace labeled **Integrity Demo** is not showing, perform the following steps to go to the demo top level workspace:

To bring up the Integrity SNMP Demo workspace:

- 1 Click on the background of the G2 window to display the G2 Main menu and choose Get Workspace.
- 2 Select OXS-DEMO-TOP-LEVEL from the named workspaces list.

The demo gives examples of how to use many of the functions and features of the knowledge base. The demo top-level workspace is divided into the following areas:

- Initializations - Initialization items for use with this demo.
- Get, Set & Send Traps - Examples of performing the various SNMP functions a G2-SNMP Bridge Heartbeat routine, and performance parameters for measuring SNMP trap input traffic.
- Simulated Traps - An example of using the simulated trap facility.
- Simulated MIBs - Examples of using the simulated MIB facility.

- Reading trapd.conf File - Examples of routines used in reading the trapd.conf file.
- Sample Code - Sample code for using the Clears For utility and sending traps to other processes.

Connecting G2 to the GSI Bridge Process

In order for Integrity to be able to perform the SNMP transactions (send/receive traps, GET, and SET) you must configure Integrity to talk to the SNMP Gateway Bridge process. The following steps outline what must be done in order for G2 and the SNMP Gateway Bridges to communicate.

Two GSI interface objects already exist in the Integrity product:

- INTEGRITY-GSI-TRAP-SENDER
- INTEGRITY-GSI-TRAP-RECEIVER

To connect G2 to the GSI process, fill out the attributes of the GSI Interface Objects, and connect G2 to the GSI Bridge processes via the GSI Interface objects.

The `oxs_demo.kb` contains examples of two GSI Interface objects for connecting to the SNMP Gateway Bridge processes.

For more information about the GSI interface objects, see the *G2 Gateway User's Guide*.

Creating a GSI Interface Object

You can create a SNMP GSI interface object by navigating to the palette:

- G2 Classic:
View > Palettes > Toolbox - G2 > NetworkInterfaces - SNMP
- TWNG:
View > Toolbox - G2, Network Interfaces - SNMP palette

Select the SNMP Interface from the palette and placing it on a workspace in your application.

Configuring the GSI Interface Object

Once the snmp-interface object has been created it is necessary to fill in the appropriate attributes.

Updating the attributes of the snmp-interface object:

- 1 Right click the snmp-interface object and choose properties.
- 2 Enter values for these properties:

Attribute	Description
Name	The interface name, which cannot contain spaces.
Bridge host	The host where the bridge is running.
Bridge port	The port number that the SNMP bridge will be listening on, for example, 22041.
Remote process initialization string	<p>The initialization string used by the SNMP Gateway Bridge to determine the mode of operation in which it will run. The following flags make up this string:</p> <p>-d: Debug ON, no -d flag specified indicates that debug is OFF.</p> <p>-t #: The timeout period for SNMP retries, in tenths of a second specified as an integer. This value is used directly by the HP OpenView SNMP API.</p> <p>-p #: Mode of operation that the SNMP Gateway Bridge operates in. Possible values are:</p> <ul style="list-style-type: none">1 - Receive Traps2 - GET, SET, and Send Traps. <p>For example:</p> <pre>-d -t 20 -p 1</pre>
Auto Reconnect To Bridge	If the interface status goes to -2 (disconnected), whether to attempt an automatic reconnect.
Shutdown Bridge Upon Disconnect	If enabled, sends a shutdown command to the bridge.
Launch Bridge Upon Connect	If enabled, launches the bridge based on the Bridge Launch Shell Script value.

Attribute	Description
Bridge Launch Shell Script	Script to launch the specified bridge for this interface.
GSI interface status	<p>Status of the connection between this G2 and the GSI bridge process, automatically updated by G2 after each transmission between G2 and the bridge process.</p> <p>Note: This attribute is read only.</p> <p>Possible values:</p> <p>-2 (Error) - An error condition occurred, and the connection has broken between G2 and the GSI bridge process.</p> <p>-1 (Timeout) - The G2 process has not heard from the bridge process within the Interface-timeout-period specified for the interface object. This code may also indicate that a communications overload has occurred.</p> <p>0 (Inactive) - The interface is either disable or inactive.</p> <p>1 (Initializing) - The external system is initializing. When G2 receives this code, it sends no further messages to the bridge process until it receives an OK code.</p> <p>2 (OK) - The connection between the G2 process and the bridge process is successful and being maintained.</p>

G2-SNMP Bridge Setup

Describes how to configure the G2-SNMP Bridge, to set up Integrity to process SNMP traps, and to perform additional processing in response to incoming traps.

Introduction	424
Configuring the G2-SNMP Bridge	425
Trap Handling Overview	428
Trap Class Creation and Processing	430
Completion Procedure Determination	441
SNMP Transactions	442
Sending Traps to External Systems	443
Simulation Facilities	445



Introduction

The following steps summarize how you set up your G2-SNMP Bridge application:

- 1 Create and configure the synchronous and asynchronous GSI interface objects. See the *G2 Reference Manual* for additional information on configuring GSI interface objects.
- 2 Customize the configuration of the G2-SNMP Bridge. For details, see [Configuring the G2-SNMP Bridge](#).
 - a Integrity. For details, see [Defining Initializations](#).
 - b SNMP Gateway Bridge (SGB). For details, see [SNMP Gateway Bridge Configuration](#).
- 3 Define the trap class definitions. For details, see [Trap Handling Overview](#) and [Trap Class Creation and Processing](#).
 - a Handling unrecognized traps. For details, see [Handling Unrecognized SNMP Traps](#).
 - b By reading vendor supplied MIBs. For details, see [Vendor MIBS](#).
 - c By reading the trapd.conf.ppd file. For details, see [trapd.conf.ppd Parser](#).
- 4 Define the Clears-For attribute. For details, see [Clears-For Attribute](#).
- 5 Define the completion procedures for trap processing. For details, see [Completion Procedure Determination](#).
- 6 Perform SNMP transactions (SET, GET, and GET NEXT) for additional processing. For details, see [SNMP Transactions](#).
- 7 Send SNMP traps to external systems for additional processing. For details, see [Sending Traps to External Systems](#).
- 8 Perform simulations for testing. For details, see [Simulation Facilities](#).
 - a Simulated traps. For details, see [SNMP Trap Simulation](#).
 - b Simulated MIBs. For details, see [SNMP Agent MIB Simulation](#).

Configuring the G2-SNMP Bridge

The G2-SNMP Bridge functions as a point to point connection. There is one SGB process for one G2 process. This means that only one G2 process can connect to an SGB Trap Receiver process. However, another SGB GET, SET, and Trap Send process can connect to the same G2 process that the SGB Trap Receiver process is connected to. The SGB Trap Receiver process was not designed to handle multiple G2 connections. To achieve this type of functionality you can run multiple SGB

Trap Receiver processes with each one connected to a separate G2 process. For each of the SGB Trap Receiver processes you can define what traps they should filter, see [Filtering Traps](#), thus dividing up the traps that are sent to each of the individual G2 processes.

SNMP Gateway Bridge Configuration

Through calls to procedures in the SNMP Gateway Bridge, your G2-SNMP Bridge application can configure parameters and filter traps.

Communication Parameters

The SNMP Gateway Bridge provides a remote procedure call for modifying a subset of its communication parameters:

- `g2snmp_modify_comm_params`

You can also use two procedures to select which communication parameters the SNMP Gateway Bridge uses:

- `g2snmp_use_snmp_defaults`
- `g2snmp_use_snmp_comm_params`

Any remote procedure declarations in your application that modify the SNMP Gateway Bridge communication parameters must specify one of these functions as the Name-in-remote-system attribute of the REMOTE PROCEDURE DECLARATION object. See the *G2 Reference Manual* for additional information on declaring remote procedure calls.

Filtering Traps

Two procedures help you manage the SNMP Gateway Bridge's filtered trap definition list:

- `g2snmp_add_filtered_trap`
- `g2snmp_delete_filtered_trap`

The SNMP Gateway Bridge can filter traps that your application does not want or need to receive. The SNMP Gateway Bridge does not send filtered traps to Integrity.

The SNMP Gateway Bridge requires a filter definition file to determine which traps to filter. The `trapd_pp` utility included with the G2-SNMP Bridges distribution reads a file in the format of the HP OpenView `/usr/OV/C/trapd.conf` file, extracts trap filtering and formatting information, and produces the filter definition file for use by the SGB. See the appropriate HP OpenView or NetView 6000 documentation or the HP OpenView *man* pages for information about the format of the `trapd.conf` file.

The filter definition file is called *filename.ppd*, where *filename* is the name of the input file. The *ppd* extension (for pre-processed) is appended to the filename. For example, *trapd.conf* becomes *trapd.conf.ppd*.

The name of the filter definition file is communicated to the SNMP Gateway Bridge through a UNIX environment variable. The environment variable `G2_PPD_FILENAME` should be set to the full path name of the filter definition file (pathname and filename). When you start the SNMP Gateway Bridge, the SNMP Gateway Bridge will read this filter definition file, as defined in the environment variable.

Note The exact commands needed to set UNIX environment variables vary widely among different workstations and different version of the UNIX operating systems. Please consult the documentation for your workstation's operating system to determine the proper commands needed to set the `G2_PPD_FILENAME` environment variable.

Telling the SNMP Gateway Bridge Which Traps to Filter

Filtered traps are specified in the *trapd.conf* file as *IGNORE* or *LOGONLY*. The traps are identified by the three-element identification vector containing enterprise ID, generic trap ID, and specific trap ID. These traps appear in the filter definition (*.ppd*) file. Every trap received by the SNMP Gateway Bridge that matches one of the specified three-element identification vectors is ignored in the SNMP Gateway Bridge, without any interaction with Integrity.

While the SNMP Gateway Bridge is running, your Integrity application can turn filtering on or off for any trap using the following remote procedure calls:

- `g2snmp_add_filtered_trap`. This remote procedure call toggles the trap ON if it is already in the filtered trap list, or adds it to the filtered trap list if it is absent. The SNMP Gateway Bridge will not pass the trap to G2.
- `g2snmp_delete_filtered_trap`. This remote procedure call toggles the trap OFF if it is present in the filtered trap list. The SNMP Gateway Bridge will now pass the trap to G2.

Caution A filter definition file *must* be specified at start-up in order to enable use of these procedures. Failure to specify a start-up filter definition file can result in failure of the SNMP Gateway Bridge process.

Error Handling

The error handling routines in Integrity use the Integrity message system if it is available. If you are not integrating with an Integrity application, then the Integrity message system will not be available and error messages will be sent to the message board. Customize this behavior by using the initialization object

`devu-error-handler-proc` to define a user application specific error handler. This error handler can redirect the error messages to some place other than the message board if desired.

Define the behavior of the default error handler by using initialization objects. The table below lists each initialization used as part of the error handling system and describes how to use it:

Initialization Item	Error Handling Definition
<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.
<code>devu-low-priority</code>	Value assigned to a low-priority error.
<code>devu-system-category</code>	Value assigned to the message category of a system error.

You can change any of the default values for these items by using the GFR startup procedures for your application.

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. These arguments are described below:

`user-defined-error-handler`

(*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

Argument	Description
<i>error-text</i>	Text describing the error
<i>error-lifetime</i>	Time interval in seconds before the error is deleted

Trap Handling Overview

Traps are received from SNMP agents by Integrity via the HP OpenView, Netview 6000, or Generic Bridges.

In general, first the trap is assembled and the incoming values are placed into a mib-receiver object. Then, the completion procedure for the trap is called, if it exists. The completion procedure is defined by the user. Refer to [Completion Procedure Determination](#) for details on defining a user defined completion procedure.

Note Integrity provides a basic message structure for representing alarms, messages, and/or events received via SNMP traps and other asynchronous events. SNMP traps received from the SNMP Gateway Bridge map directly into the message attributes (sender object, target object, message ID, creation-time, etc.) of the Integrity message. In addition to the textual representation of the message, Integrity sets relations to the sending objects, target objects, etc.

If you are not integrating the G2-SNMP Bridge with an Integrity application, then it will be necessary for you to develop additional software, as needed, for the processing of an SNMP trap.

Values within a single trap are passed separately as object identifier (OID)/value pairs and assembled in Integrity into a mib-receiver object. The OID identifies a piece of data defined in the SNMP device's MIB database. The value of that database field is the second half of the OID/Value pair. The OID's to be passed with a specific trap are defined by the TRAP-TYPE declaration statements in the ASN.1 formatted MIB definition file for the SNMP device. Integrity should have already read and processed these definitions before the trap is sent, although this is not required. Refer to [MIB Processing](#) for details.

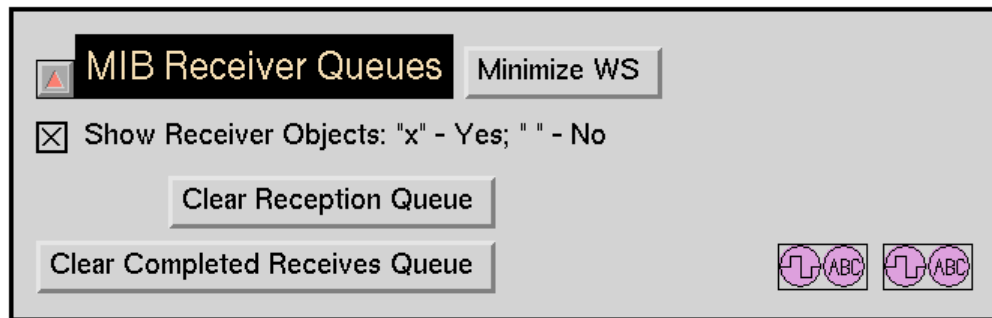
When receiving an SNMP trap, Integrity must determine which class of mib-receiver object is appropriate for storing and managing the data being passed with the trap. After instantiating the appropriate receiver object and storing the incoming values in the attributes of that object, Integrity next looks for a completion procedure to execute. In general, the completion procedure provides a software-hook for any application specific trap handling.

You may wish to check the values of the traps being received into Integrity. The MIB-RECEIVER objects contain the trap information and are stored in the mib

receiver queues. In addition, you may also want to occasionally clear those queues.

To view the MIB Receiver Queues workspace:


- 1 Choose Workspace > Get and select gmib-top-level.
- 2 Select Programmer's Interface, then MIB Receiver Queue.



To clear the queues:

- Click the Clear Reception Queue and/or Clear Completed Receives Queue buttons.

The following figure is an example of a MIB-RECEIVER object for an SNMP trap:



a snmp-simulated-trap-receiver	
Notes	OK
Item configuration	none
Names	none
Result id	-99999
Number of received values	1
Number of expected values	1
Number of errors	0
Creation time	15 Aug 97 12:25:23 p.m.
Receive completion method	oxs-demo-completion-procedure
Description	"trap"
Error code	-99999
Error string	""
Enterprise id	"openView"
Agent address	"199.93.54.65"
Agent hostname	"oxs-demo-node"
Generic trap	6
Specific trap	"56789"
Agent act time	-99999
Default severity	1
Default category	"hpopenview"
Results list	a text-list
Trapd format	"Enterprise \$E Generic \$G Specific \$S"
Clears for	""

Trap Class Creation and Processing

Several key values are always provided by the SNMP Gateway Bridge as part of every trap received. These include:

- The enterprise ID of the sending agent
- The generic trap number (0-6)
- The specific trap number (only meaningful if the generic trap is 6)

In determining the class of MIB-RECEIVER object to create for an incoming trap, Integrity searches for class definitions in the following order:

- 1 TRAP-[enterprise-id]-[generic-id]-[specific-id]
For example, TRAP-WELLFLEET-6-57789382
- 2 TRAP-[enterprise-id]-[generic-id]-xx
- 3 TRAP-[enterprise-id]-xx-xx

- 4 TRAP-xx-[generic-id]-xx
- 5 TRAP-xx-xx-xx

Note Class definition names can use **xx** in place of the enterprise-id, generic-id or specific-id. Integrity will match **xx** with any value from the trap for the enterprise-id, generic-id or specific-id.

Caution If a trap receiver class definition named TRAP-XX-XX-XX (item #5) exists, it will be instantiated for *any* enterprise-id/generic-id/specific-id trap combination *unless* a more specific class definition (items #1 - #4) exists.

Handling Unrecognized SNMP Traps

An SNMP trap received by Integrity with no corresponding trap class definition, TRAP-[ENTERPRISE-ID]-[GENERIC-ID]-[SPECIFIC-ID], can have a trap class definition created automatically and placed upon a storage workspace. Setting the initialization parameter `mib-create-nonexistent-traps` to a value of 1 will cause Integrity to automatically create the trap class definition. See `mib-create-nonexistent-traps` for more information on setting this value. The default value for `mib-create-nonexistent-traps` is 0, which means *do not create* unrecognized SNMP traps. The following diagram shows the workspace where unrecognized trap class definitions are placed. The initialization parameters `oxs-default-unrecognized-trap-class` and `oxs-unrecognized-traps-location` can be customized by the user to control the behavior of the creation and placement of unrecognized SNMP traps

To view the Integrity Unrecognized Traps workspace:

- 1 Click on the background of the G2 window and choose Get Workspace from the G2 Main menu.
- 2 Select OXS-SIMULATION-TOP-LEVEL from the named workspaces menu to display the workspace.
- 3 Click the Unrecognized Traps button to display the following workspace:



SNMP Traps

With Integrity 4.0, a trap manager and trap properties were introduced and is still present in the current release of Integrity. This allows all received traps to be processed by a single completion procedure making the handling of traps easier. This is accomplished by setting the Use SNMP Java Bridge checkbox on the SNMP Setup tab of the Setup dialog. If this is not set, trap processing is handled by the process described in the previous section.

Trap Manager

There is only one trap manager per G2 process. The trap manager keeps track of all traps received by the application in a queue. The trap manager has the ability to create new trap properties if one does not exist for the new trap. It also creates a new domain object if the domain object does not exist in the current application.

Trap Properties

The class `gmib-trap-properties` encapsulates a single trap. You can access trap properties for the application, in the Navigator, by selecting System Models > SNMP Traps. You can create new trap properties in one of two ways: first, in the Navigator, by right-clicking the SNMP Traps folder and choosing New Instance, or second, from the Toolbox - SNMP Traps toolbox.

To view or modify a trap property, right click on the trap property object or right click on the trap property name within the Navigator and choose Properties.

Here is a description of the properties:

Trap Property	Description
Trap OID	OID value in the ASN1 format.
Trap Named OID	The named OID of the trap.
Short Name	The label of the trap.
Trap Variable Labels	The MIB object names of the trap.
Description	Description of the trap.
Trapd Format	The format specification for the trap message processed by the trapd parser (located in the Setup dialog).
Custom Message	A customer message for the trap.
Passport Stamps	Passport stamps used by ODIE.

Trap Property	Description
Trap Class	The trap receiver class to instantiate when a trap is received by the application.
Signature	A two pair value representing the default trap signature from the MIB file and the short name of the trap property.
Ignore Trap	If true, no processing will be done for the trap.
Severity	Severity for the trap.

Defined Trap Properties

Integrity supplies six SNMPv1 trap properties, which provides Integrity with the ability to process these traps automatically with no configuration. These v1 traps include to following:

- Cold Start
- Warm Start
- Link Down
- Link Up
- Authentication Failure
- EGP Neighbor Loss

You can access these trap properties in the Navigator by selecting System Models > SNMP Traps. Any trap property created by MIB parsing or by the Trapd processing are displayed here as well.

Trap Processing

When a trap event is received through the SNMP bridge, SNMP version-specific information is processed first, then a generic approach to process the rest of the information is performed. The generic processing includes locating the trap properties based on the trap OID. If no trap property exists, one is created with minimal information. Once a trap property is identified, a trap receiver object is created. The trap receiver object contains all of the trap information, along with any OID value pair associated with the trap event. If the trap event is to clear an existing trap (i.e., the Link Up trap clears the Link Down trap), it is processed before generating a `gtrap-trap-receive-created` event. This event is dispatched using the GRTL event scheme (see the GRTL documentation for information).

An event listener defined in the GTRAP module listens for `gtrap-trap-receiver-created` events. When this event is generated, the `oxs2-trap-completion` method is called. This is a generic completion procedure used for all trap events. The

completion procedure locates the domain object based on the trap event information. If no domain object is found and the trap monitor is configured to create new domain objects, a new domain object is created for the trap event. A message is then created and posted to the message browser containing the information in the trap event. When using ODiE, an event is generated for ODiE as well.

If other trap processing is needed, applications can define their own event listener and listen for the `gtrap-trap-receiver-created` event.

MIB Processing

The Integrity MIB parser is a combination of a Java application and a KB module. It allows you to import MIBs for the equipment you want to monitor and manage. The import is based on the AdventNet libraries. If the MIB you are importing requires other MIB files, the parser also parses those files as well.

Setting Up and Running the MIB Parser

The startup and shutdown of the bridge is all controlled from the Integrity application within the Setup dialog. You can activate the setup dialog by first changing to Developer mode and selecting the Setup icon within the Integrity toolbar. The dialog is divided into two sections: MIB Parser Setup and MIB Processing.

MIB Parser Setup

The Setup dialog controls the startup and shutdown of the MIB Parser process. The Parser text box contains the absolute path, including the startup batch file, to launch the parser. If the location has changed, select the "..." button to launch the File Select dialog to respecify the location and startup file.

To launch the parser, click the Launch Parser button, which executes the `StartJMibParser.bat` file. A notification dialog appears while the bridge is attempting to start. It disappears when the bridge makes a connection or gives up due to a timeout condition. The first time you start the parser, it may take some time as the Java VM must load as well. Subsequent launching of the parser will be faster. The disabled text box to the right of the Launch Parser button displays the current connection status of the parser.

To shutdown the parser, click the Shutdown Parser button. This will kill the Java process, and any further attempts to process MIB files will be unsuccessful.

Processing MIB Files

To select a MIB file for processing, click the "..." button next to the File text box. If the MIB file you are processing requires other MIB files, the parser parses those files as well, as long as they are contained in the same directory as the original selected file.

Once the MIB file has been specified, click the Process MIB button to begin parsing the file. The parsing process includes the creation of a `gmib-mib-reader` object. This object contains default settings, parsed information, and the resulting objects created during the parsing. The `gmib-mib-reader` object is added to the Navigator under System Models > Parsed MIBs. From here, you can display the properties of the `gmib-mib-reader` object and view all of the objects created during the parsing of the file, as well as display and modify the properties of these objects.

Viewing a Parsed MIB

To view the parsed MIB files, in the Navigator, select System Models > Parsed MIBs. A list of MIB files that have already been parsed for the application appears. To view the OID to name translation objects created for the MIB, click the "+" next to the MIB name. The tree expands showing a list of all OID to name translation objects.

To view or modify the properties of an OID to name translation object, right click the OID and choose Properties.

To view or modify the properties of the MIB reader object, right click the `gmib-mib-read` object and choose Properties.

Installed MIBs

Integrity provides the following MIB files already parsed using the MIB parser:

Installed Parsed MIBs

UPS B Enterprise

Cisco Enterprise OIDs

Cisco OIDs

Controll Enterprises

HP UNIX Enterprises

HP UNIX OIDs

HP UNIX Special Enterprise Entries

ODS 290 ENC Enterprises

ODS 290 FDDI 7-3 Enterprises

ODS 290 FDDI Enterprises

ODS 290 Inf Agent Enterprises

Installed Parsed MIBs

ODS 290 Inf Enterprises
ODS 290 Inf Tr Enterprises
ODS Inf Chassis Enterprises
ODS Inf FDDI Enterprises
RFC1213 Enterprises
RFC1213 OIDs
RMON OIDs
Telenex Enterprises
Wellfleet Enterprises
Wellfleet OIDs

Vendor MIBS

Management Information Bases (MIBs) are available in the public domain, generally supplied by vendors. To save space, you only need to read in the MIBs relevant to your site and you should delete the ones you don't need. Some MIBs are provided as a part of Integrity, such as those contained in the *asn1.kb* module. You can merge this module into your application and make use of those MIBs. See the *G2 Reference Manual* for information on merging modules into a knowledge base.

Caution Any modifications done directly in the *gmib.kb* module could potentially be lost when you install a new version of Integrity. To avoid this situation, transfer the navigation buttons of the desired MIB from the *gmib.kb* top-level workspace to a location in your application.

To go to the GMIB top-level workspace:

- 1 Click on the background of the G2 window to display the G2 Main menu and choose Get Workspace.
- 2 Choose GMIB-TOP-LEVEL from the named workspaces menu, select Programmer's Interface, then Enterprise OID to Name Translations to display the following workspace:



trapd.conf.ppd Parser

The file *trapd.conf* is supplied with HP OpenView as well as NetView 6000. This file is the only place where information related to the severity (priority) of traps and the format of their translation to printed log messages is kept.

Integrity provides a facility for reading this file and creating trap class definitions for the various Enterprise ID event definitions. In order for Integrity to make use of this file the C program *trapd_pp*, supplied as part of the G2-SNMP Bridges, must be executed to pre-process the contents of the file *trapd.conf*. The output of this program is the file named *trapd.conf.ppd*. The original *trapd.conf* file is not changed during the pre-processing.

Note Enterprise ID's are also contained in this file. However, this information is assumed to exist elsewhere and is ignored.

Use the following procedure for reading in the pre-processed *trapd.conf.ppd* file:

```
mib-trapd-preprocessed-conf-reader
(filename: text, logfile: text, ws: class kb-workspace,
modify-existing: truth-value)
```

Argument	Description
<i>filename</i>	The pathname of the <i>trapd.conf.ppd</i> file to be read
<i>logfile</i>	The pathname of the file that logs the results of the read process.
<i>ws</i>	A kb-workspace where the newly created trap class definitions are to be placed.
<i>modify-existing</i>	Determines whether the procedure should modify existing trap class definitions.

Caution Unknown Enterprise IDs will be created as *transient* OID-TO-NAME-TRANSLATIONS objects with a resulting name of “Unknown Enterprise”.

The *mib-trapd-preprocessed-conf-reader* procedure processes the *trapd.conf.ppd* file as follows:

- 1 Reads down to the Enterprise ID entries.
- 2 Reads the Enterprise ID entries to find the ones that are the beginning of an event definition.
- 3 For each Enterprise ID event definition, determine if a corresponding trap definition, MIB-RECEIVER class, exists by concatenating TRAP-[ENTERPRISE-ID]-[GENERIC-ID]-[SPECIFIC-ID].
 - If the trap definition does not exist then:

Check to see if the same specific trap exists for the immediate parent of the Enterprise ID (i.e. remove the last element of the Enterprise IDs dot notation).
 - If the immediate parent does not have the trap defined for that Enterprise ID then:

Create an SNMP-UNRECOGNIZED-TRAP subclass definition and place it on the kb-workspace defined by *ws*. This definition will allow the trap to be received by Integrity and any number of fields to be passed. It will also provide a place to put the default severity information from the *trapd.conf.ppd* file.
- 4 Retrieves the severity (*sev*) from the 7th field of the Enterprise ID event definition and changes the text of the attributes specific to the class of the corresponding MIB-RECEIVER class definition to include; Default-severity is a integer, initially is [*sev*];

The following function is used to convert *trapd.conf.ppd* severity values to Integrity severity values:

```
mib-default-trapd-priority-conversion
  (input-value: integer)
  -> priority: integer
```

This procedure returns an integer priority value (as defined in Integrity) converted from the severity *input-value* (as defined in HP OpenView), as follows:

<i>input-value</i>	<i>priority</i>
1	6
2	4
3	3
4	2
5	1
any value	1

- 5 Continues processing Enterprise ID event definitions until the end of file.

Clears-For Attribute

In some systems, such as Wellfleet routers, traps are sent which are intended to clear earlier traps. Clears-for means that the present incoming trap has cleared a previously received trap condition. The information linking the original trap and the traps which clears-for the original trap is missing in the ASN.1 MIB definition supplied by Wellfleet. Integrity can read and write files which capture the relations between these traps. See *mib-read-clears-file* and *mib-write-clears-file* for additional information on the use of these procedures. Unfortunately, the manufacturers do not always publish the clears-for information, so it may have to be entered by hand.

The developer is responsible for writing the completion procedure to process the clears-for entries. An example procedure, *sample-process-all-clears-for-entries* (...), is provided in the *oxs_demo.kb* knowledge base. This example procedure assumes that the G2-SNMP Bridge is integrated with an Integrity application.

To manually enter a clears-for attribute

- 1 Click the trap class definition to be edited and choose table from its menu.

The following figure is an example of trap class definition attributes table:



TRAP-WFSWSERIES7-6-789545, an object-definition	
Notes	OK
Authors	dgreen (11 Sep 1997 4:15 p.m.), hill, dave
Change log	0 entries
Item configuration	none
Class name	trap-wfswwseries7-6-789545
Direct superior classes	trap-xxxx-xx-xx
Class specific attributes	none
Instance configuration	none
Change	none
Instantiate	yes
Include in menus	yes
Class inheritance path	trap-wfswwseries7-6-789545, trap-xxxx-xx-xx, snmp-trap-receiver, mib-receiver, object, item
Inherited attributes	trapd-format is a text, initially is "Default Trap format for Ent: \$E Generic \$G Specific \$S (\$#):\$*"; description is a text, initially is "trap"; error-code is an integer, initially is -99999; error-string is a text, initially is ""; enterprise-id is a text, initially is ""; agent-address is a text, initially is ""; agent-hostname is a text, initially is ""; generic-trap is an integer, initially is -99999; specific-trap is a text, initially is "-99999"; agent-act-time is an integer, initially is -99999; default-severity is an integer, initially is 5; default-category is a text, initially is "noformat"; results-list initially is an instance of a text-list; result-id is an integer, initially is -99999; number-of-received-values is an integer, initially is 0; number-of-expected-values is an integer, initially is 99999; number-of-errors is an integer, initially is 0; creation-time initially is an instance of a mib-time-parm; receive-completion-method is a symbol, initially is no-method
Attribute initializations	none
Icon description	inherited
Attribute displays	inherited
Stubs	inherited

- 2 Edit the Class-specific-attributes attribute of the object definition.

3 Enter the new attribute in one of the following formats:

- clears-for is a text, initially is “<enterprise-id>-<generic-id>-<specific-id>”

For example:

clears-for is a text, initially is “HPOpenView-6-912875”

This specifies that the reception of this trap will clear the HPOpenView-6-912875 trap condition.

- clears-for is a text, initially is “<enterprise-id1>-<generic-id1>-<specific-id1>, <enterprise-id2>-<generic-id2>-<specific-id2>, ... <enterprise-idn>-<generic-idn>-<specific-idn>,”

For example:

clears-for is a text, initially is “912875, 874356, 235681”

Completion Procedure Determination

In determining the completion procedure to execute for an SNMP trap, Integrity searches for procedure definitions in the same order it uses for MIB-RECEIVER class definitions. This value is specified in the Receive-completion-method attribute of the MIB-RECEIVER object. This allows an application specific default to be provided.

The order of the search is as follows:

1 completion-[enterprise-id]-[generic-id]-[specific-id]

For example, completion-wellfleet-6-57789382

2 completion-[enterprise-id]-[generic-id]-xx

3 completion-[enterprise-id]-xx-xx

4 completion-xx-[generic-id]-xx

5 completion-xx-xx-xx

Note Completion procedure names can use xx in place of the enterprise-id, generic-id or specific-id. Integrity will match xx with any value from the trap for the enterprise-id, generic-id or specific-id.

Caution If a completion procedure named completion-xx-xx-xx (item #5) exists, it will be executed for *any* enterprise-id / generic-id / specific-id trap combination *unless* a more specific procedure (items #1 - #4) exists.

To see a sample completion procedure:

- G2 Classic
View > Palettes > Toolbox - SNMP Traps > SNMP Trap Processing
- Telewindows
View > Toolbox - SNMP Traps > SNMP Trap Processing

SNMP Transactions

Through calls to procedures in the SNMP Gateway Bridge, your G2-SNMP Bridge application can perform both blocking and non-blocking transaction requests. Normally, you will use non-blocking calls, so that Integrity and the SGB can both work on other tasks while waiting for responses to requests. This means you are normally operating in a call back manner.

For instance, to do an SNMP GET, you issue a call to the SGB, and the SGB issues you a unique identifier. When the response comes back from the network, the SGB calls receiver procedures in Integrity. One call is made for each attribute. The values returned in these calls are assembled by Integrity into objects called MIB-RECEIVERS. The MIB-RECEIVER is then placed in the MIB receiver queue.

The `oxs_sim-request-handler` procedure handles the simulated and real time SNMP transactions (SET, GET, and GET NEXT) of your G2-SNMP Bridge application. See `oxs_sim-request-handler` for information on the argument list and return values of this procedure.

Integrity acquires data from the SNMP Gateway Bridge through:

- Return values of the SNMP Gateway Bridge RPC's (GET, GET NEXT). Calls to remote procedures in the SNMP Gateway Bridge may be blocking or non-blocking.
- Values that the SNMP Gateway Bridge returns to the Integrity receiver RPC's (Traps), through non-blocking RPC's.

Caution Integer values in G2 are signed with 30-bit precision. A G2 integer value can range from -2^{29} to $(2^{29} - 1)$. For values outside of this range the SNMP Gateway Bridge will return the value as a float, see the *G2 Reference Manual* for more information.

Blocking and Non-Blocking Transactions

The SNMP Gateway Bridge provides two SNMP transaction procedures you access from Integrity called:

- `g2snmp_blocking_transaction`
- `g2snmp_nonblocking_transaction`

These procedures handle any SNMP requests from your G2 application. Any remote procedure declarations in your G2 application that use SNMP transactions must specify one of these two functions as the `Name-in-remote-system` attribute of the `REMOTE-PROCEDURE-DECLARATION` object. For descriptions of these procedures, see [Core Services APIs](#). See also the *G2 Reference Manual* for additional information on defining Remote Procedure Calls.

Overloading Remote Procedures

The GSI remote procedure call feature allows overloading of G2-to-GSI RPC's. This means that several different G2 remote procedure declarations can reference a single GSI function in the SNMP Gateway Bridge. This enables a variable number and variety of types of arguments to be passed to the GSI function.

This overloading capability is an essential feature in the use of the two G2-SNMP Bridge transaction procedures, blocking and non-blocking.

Sending Traps to External Systems

HP OpenView Interface

Integrity can use traps to send messages directly to an HP OpenView window. For instance, a common need is to change the color of a status display on the HP OpenView window. This can be done using the standard status change trap.

Sending an HP OpenView status trap

A status trap changes the status, and hence the color, of an object in HP OpenView. In order to effect a color change in HP OpenView the status source of the object must be set to "OBJECT". Note that you can access an object by name, such as "localhost", or you can address it hierarchically, e.g., "localhost.lan0" for its ethernet interface.

An equivalent method of setting the status, which can be found in the HP OpenView *man* pages, is:

```
/usr/OV/bin/  
localhost  
  
.1.3.6.1.4.1.11.2.3.2.3 "" 6 58916871 ""  
  
.1.3.6.1.4.1.11.2.15.2.0 integer 14  
  
.1.3.6.1.4.1.11.2.15.3.0 octetstring $OBJECT  
  
.1.3.6.1.4.1.11.2.15.4.0 octetstring  
  
Object status is  
  
.1.3.6.1.4.1.11.2.15.5.0 octetstring $NEWSTATUS
```

You should create this script, so that you can test the trap-receive procedures independently of their generation in Integrity, and for a convenient way to reset the color in HP OpenView following a change from Integrity. If you write a script to do this, put a backslash “\” at the end of each line except the last (the backslash character in a shell script denotes continue at next line).

The *oxs_demo.kb* knowledge base provides an example of a send status procedure, **sample-send-ov-trap**.

This procedure assumes you have defined the environment variables *\$OBJECT* and *\$NEWSTATUS*. Of course, you can directly put object names in quotes in place of *\$OBJECT*, and do the same with the possible status values.

The possible status values are "Normal", "Critical", "Marginal", and "Unknown". The HP OpenView manual also specifies that you can use "Up" for Normal, and "Down" for Critical. Spelling, spaces, and capitalization must be exact, or HP OpenView will not understand it. See the HP OpenView *trapd.conf man* pages for more details.

Note You can follow these changes in the log */usr/OV/log/trapd.log* as well as watching the HP OpenView window map. You will see errors in format, etc., much better in the log.

NetView 6000 Interface

The interface to IBM's NetView 6000 is essentially the same as the HP OpenView interface. See [HP OpenView Interface](#) for more details.

Simulation Facilities

Integrity provides facilities for sending simulated SNMP traps and for getting variables from a simulated MIB. For examples of these facilities see the *oxs_demo.kb* knowledge base.

SNMP Trap Simulation



Integrity provides an object, SNMP-SIMULATED-TRAP-RECEIVER, for filling in simulated trap information and the procedure `oxs-sim-simulate-trap (...)` for sending a simulated trap to Integrity. In addition to being able to start a simulated trap programmatically, Integrity provides a menu choice, `simulate trap`, for starting a simulated trap. The completion procedure of a simulated trap can be run automatically by selecting the menu choice `run completion procedure` from the menu choices of the SNMP-SIMULATED-TRAP-RECEIVER object.

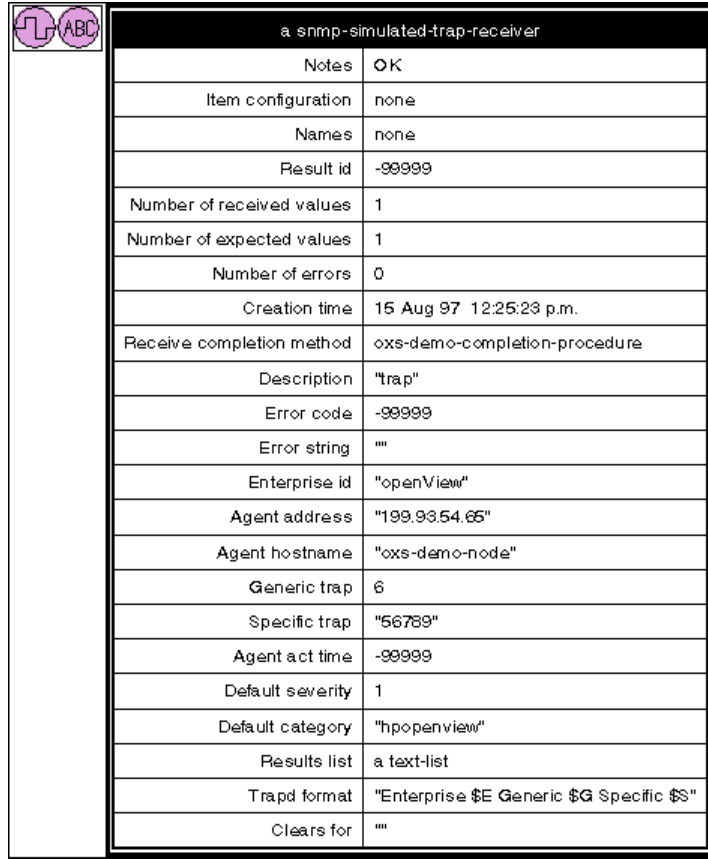
Before executing a simulated SNMP trap you will need to configure it.

To configure a simulated SNMP trap object:

- 1 Click the background of the G2 window and choose Get Workspace from the G2 Main menu.
- 2 Select OXS-SIMULATION-TOP-LEVEL from the named workspaces menu to display the workspace.
- 3 Click the Simulated Trap object.
An instance of the object is attached to the mouse.
- 4 Drop the object on the desired workspace.

- Click the object and choose table from its menu.

The following figure shows an example of the Simulated Trap attributes table:



a snmp-simulated-trap-receiver	
Notes	OK
Item configuration	none
Names	none
Result id	-99999
Number of received values	1
Number of expected values	1
Number of errors	0
Creation time	15 Aug 97 12:25:23 p.m.
Receive completion method	oxs-demo-completion-procedure
Description	"trap"
Error code	-99999
Error string	""
Enterprise id	"openView"
Agent address	"199.93.54.65"
Agent hostname	"oxs-demo-node"
Generic trap	6
Specific trap	"56789"
Agent act time	-99999
Default severity	1
Default category	"hpopenview"
Results list	a text-list
Trapd format	"Enterprise \$E Generic \$G Specific \$S"
Clears for	""

You can configure the SNMP-SIMULATED-TRAP-RECEIVER to execute the simulated trap only and/or to execute the completion procedure only.

- You must specify values for these attributes to execute the simulation of a trap:

Attribute	Description
enterprise-id	The enterprise of the trap being sent. For example: HP OpenView
agent-address	The IP address of the agent sending the trap. For example: 165.50.61.1

Attribute	Description
agent-hostname	The hostname of the agent sending the trap. For example: Gensym-1
generic-trap	The generic trap id, possible values are: 0 (coldStart) 1 (warmStart) 2 (linkDown) 3 (linkUp) 4 (authenticationFailure) 5 (egpNeighborLoss) 6 (enterpriseSpecific)
specific-trap	The specific trap id, usually vendor specific.

- 7 You must specify values for the following attributes to execute the completion procedure of a simulated trap:

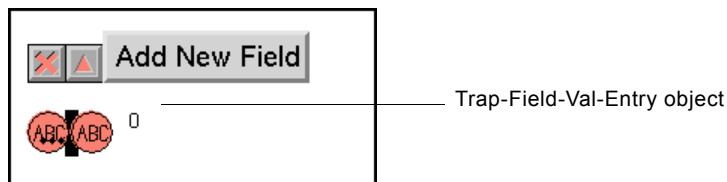
Attribute	Description
receive-completion-method	The name of the procedure to be executed after the reception of the trap. The Receive-completion-method attribute follows the format: completion-[enterprise-id]-[generic-id]-[specific-id]
enterprise-id	The enterprise of the trap being sent. For example: HP OpenView
agent-address	The IP address of the agent sending the trap. For example: 165.50.61.1
agent-hostname	The hostname of the agent sending the trap. For example: Gensym-1

Attribute	Description
generic-trap	The generic trap id, possible values are: 0 (coldStart) 1 (warmStart) 2 (linkDown) 3 (linkUp) 4 (authenticationFailure) 5 (egpNeighborLoss) 6 (enterpriseSpecific)
default-category	A text attribute for specifying the category of the trap.
trap-format	“Enterprise \$E Generic \$G Specific \$S”
clears-for	For information on this attribute, see Clears-For Attribute .
specific-trap	The specific trap id, usually vendor specific.
default-severity	The severity level of the trap as an integer value.

Many of the SNMP traps have variable values (OID/Value pairs) associated with them. Integrity provides a way for the user to specify the variable values of a simulated trap.

Inputting simulated trap variable values:

- 1 Click the SNMP-SIMULATED-TRAP-RECEIVER object and choose Go To Subworkspace from its menu.



- 2 Click TRAP-FIELD-VAL-ENTRY object and choose Table from its menu.
- 3 Enter values for these attributes:

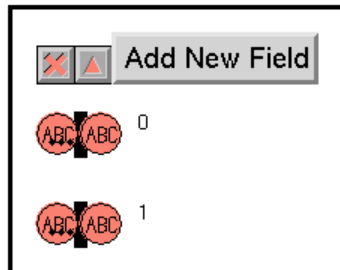
oid Dotted IP notation.
oid-val Value for the OID.

Note The field-number attribute is updated automatically.

To add a new variable:

- 1 Click the Add New Field button at the top of the workspace.

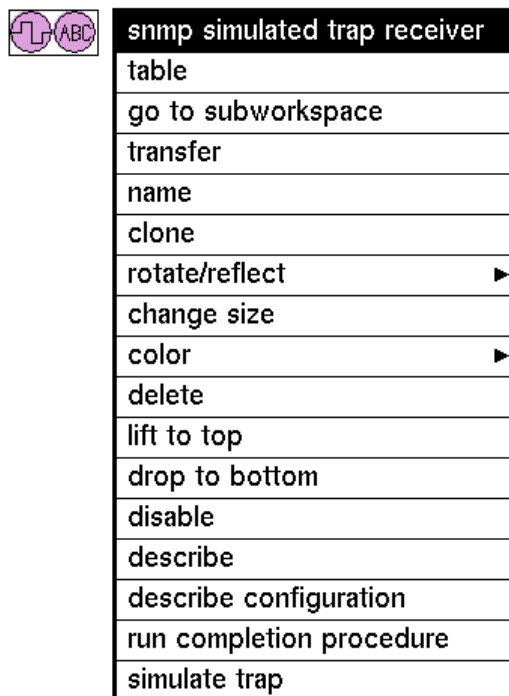
A new TRAP-FIELD-VAL-ENTRY object appears on the workspace, for example:



To execute a simulated SNMP trap

- 1 Click the SNMP-SIMULATED-TRAP-RECEIVER object to display its menu.

For example:



2 Choose one of the following two menu options:

simulate trap	Simulates the reception of an SNMP trap from the SNMP Gateway Bridge.
run completion procedure	Executes the completion procedure of the simulated trap with no simulation of the reception of the trap via the SNMP Gateway Bridge.

Simulated traps can be executed programmatically by using the procedure `oxs-sim-simulate-trap` (`mib-rec: class snmp-simulated-trap-receiver`). Selecting the menu choice `simulate trap` or executing the procedure `oxs-sim-simulate-trap` result in the same set of actions being performed.

SNMP Agent MIB Simulation



Integrity provides a facility for simulating the GET and GET NEXT SNMP transactions on agent MIBs. The `oxs_demo.kb` provides examples of how this facility is used. Integrity provides the `oxs_sim-request-handler` procedure for interacting with simulated MIBs.

To configure a simulated MIB:

- 1 Click on the background of the G2 window and choose Get Workspace from the G2 Main Menu.
- 2 Select OXS-SIMULATION-TOP-LEVEL from the named workspaces menu to display the workspace.
- 3 Click the Simulated MIB object.

An instance of the object is attached to the mouse.

- 4 Drop the object on the desired workspace.

The `Names` attribute of the `SIMULATED-AGENT-MIB` is optional but the `Reference-object` attribute *must* be filled out. The `Reference-object` attribute matches the `opfo-external-name` attribute of some `OPFO-DOMAIN-OBJECT` instance. The `OPFO-DOMAIN-OBJECT` is the host of the simulated agent MIB.

The following figure shows an example of a SIMULATED-AGENT-MIB and an OPFO-DOMAIN-OBJECT whose attributes have been filled in:

OXS-DEMO-SIMULATED-MIB1, a simulated-agent-mib		an oxs-demo-ip-card	
Notes	OK	Opfo external name	"oxs-demo-ip-card1"
Item configuration	none	_opfo highest message priority	99999
Names	OXS-DEMO-SIMULATED-MIB1	_opfo acknowledgement status	acknowledged
Reference object	"oxs-demo-ip-card1"	Address	"199.93.47.58"
		Agent	"snmp"

Once the simulated MIB has been configured you will need to set up the variables and tables of the MIB. The following steps describe this process:

To configure the variables of the simulated MIB:

- 1 Click the SIMULATED-AGENT-MIB object and choose Go To Subworkspace from its menu to display the following workspace:



- 2 Click the Create Variable button.

The workspace is updated. For example:

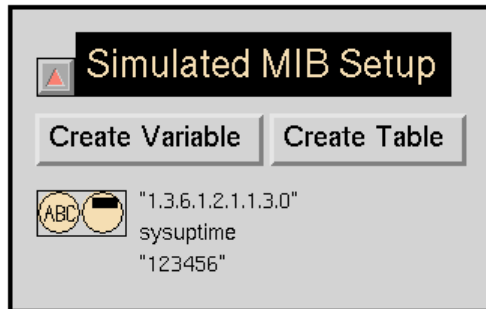


- 3 Click the MIB-FIELD-VAR object and choose Table from its menu.

4 Enter values for these attributes:

Oid	Dotted IP notation of MIB variable.
Resulting-name	The text equivalent of the OID value.
Instance-val	The value to be returned for a simulated GET.

This is the result:



Continue selecting the Create Variable action button to add additional MIB variables.

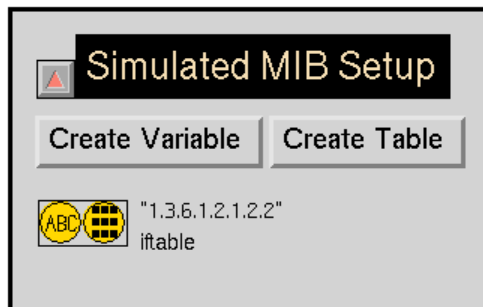
To configure the tables of the simulated MIB:

- 1 Click the SIMULATED-AGENT-MIB object and choose Go To Subworkspace from its menu to display the following workspace:



- 2 Click the Create Table button.

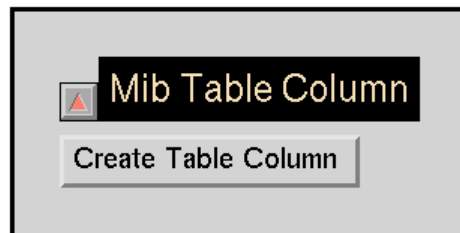
The workspace is updated. For example:



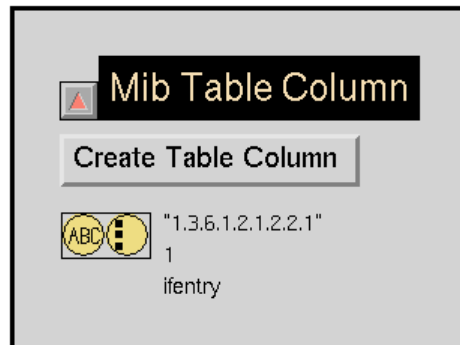
- 3 Click the MIB-FIELD-TABLE object and choose Table from its menu.
- 4 Enter values for these attributes:

oid	Dotted IP notation of MIB table.
resulting-name	The text equivalent of the OID value.

- 5 Click the MIB-FIELD-TABLE object and choose Go To Subworkspace from its menu to display the following workspace:



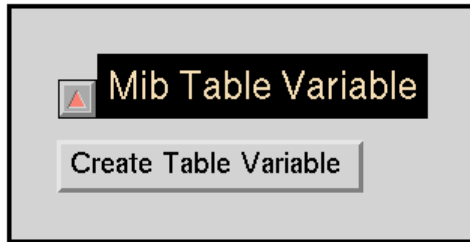
- 6 Click the Create Table Column button.
The workspace is updated. For example:



- 7 Click the MIB-FIELD-TABLE-COLUMN object and choose Table from its menu.
- 8 Enter values for these attributes:

oid	Dotted IP notation of MIB variable.
index	The index of the entry in the table.
resulting-name	The text equivalent of the OID value.

- 9 Click the MIB-FIELD-TABLE-COLUMN object and choose Go To Subworkspace from its menu to display its workspace:



- 10 Click the Create Table Variable button, and see [To configure the variables of the simulated MIB:](#) to finish the Table Variable creation.

To create additional MIB tables, return to Step 1 of [To configure the tables of the simulated MIB:](#).

G2-SNMP Bridges API

Provides a listing of the G2-SNMP Bridges APIs, remote procedure calls, procedures, and functions.

Introduction **456**

Update for GSI-Based Bridge Process **456**

Remote Procedure Calls **458**

Procedures Listed by Module **488**

Functions **503**



Introduction

This chapter provides a listing of the G2-SNMP Bridge remote procedure calls, procedures, and functions accessible to the developer.

Update for GSI-Based Bridge Process

Support for Filtering of Traps from Specified Hosts

This version of the G2-SNMP bridge supports filtering of incoming traps based on the agent hostname and/or agent IP address bound with the incoming trap. This capability is in addition to the existing filtering capability based on enterprise, generic, and specific ID's for the trap.

The following capabilities have been added to the G2-SNMP bridge:

- Those hosts for which pass or block filtering is desired are specified in a separate ASCII text file (the filtered hosts file) that contains the hostnames or the IP addresses of the hosts to be filtered. This file is specified to the bridge through the '-h <host filename>' command line option. No modifications to existing *trapd.conf* or *trapd.conf.ppd* files are required.
- The filtered hosts file may contain either a hostname or an IP address for a particular host. NOTE: Since the identification of the source host in the trap PDU is based upon the IP address of the source host, all hostnames in the filtered hosts file are mapped to IP addresses internally within the bridge. Therefore, hostnames which cannot be identified by the *gethostbyname()* Unix library function are not filtered. The file contains one filtered host identification per line. The hostname must be a string formatted exactly as it is used by the *gethostbyname()* Unix library function. All characters, including whitespace characters, from the start of the line to the carriage return, are used for matching the hostname. IP addresses must contain no whitespace characters from the initial decimal point (.) to the final digit in the address. A sample filtered hosts file might be:

```
hou38
eeyore
199.93.147.51
hou50
```

- The hostname/IP address filtering comprises only the first stage in the filtering process. Once the hostname/IP address filtering has been applied, the second stage enterprise/generic/specific filtering is applied. (See next item for details).

- The G2SNMP_SET_AGENT_FILTER_MODE RPC allows a G2 application to specify one of three states for use of the filtered host identification file:
 - **Pass:** pass only traps from the hosts specified in the file, excluding (rejecting) traps from all other hosts. Enterprise/generic/specific filtering is applied to those traps passed.
 - **Block:** block (reject) all traps from the hosts specified in the file, allowing traps from all other hosts to pass. Enterprise/generic/specific filtering is applied to those traps passed. This is the default setting for hostname/IP address filtering.
 - **Off:** bypass hostname/IP address filtering. Only enterprise/generic/specific filtering is applied to incoming traps.

See the description of the G2SNMP_SET_AGENT_FILTER_MODE RPC in the "Update for Integrity or G2 Applications" section of this document for guidelines on constructing the required RPC.

Passing Variable Values for Variable Bindings in Which the Variable Type Is 'Object Identifier'

This release of the G2-SNMP HP OpenView bridge under HP-UX passes the correct variable values for variable bindings in which the variable type is 'object identifier'. The bug fix is restricted to the GSI-based portion of the bridge. It does not affect the Integrity support KB. Previously, the bridge passed either a null string (") or a random value from memory (such as an IP address) in place of the proper value of the variable. The bridge now correctly passes varbind variable values of type 'object identifier' .

Remote Procedure Calls

The following is a list of the remote procedure calls from Integrity to the SNMP Gateway Bridge. These RPCs are:

- blocking and non-blocking transaction requests
- configuration of SNMP Gateway Bridge parameters
- addition and deletion of trap filters.

Base RPCs

The following remote procedure calls are the base RPC's used by the G2-SNMP Bridges.

g2snmp_add_filtered-trap

g2snmp_add_filtered-trap

(*enterprise-id*: text, *generic-trap-id*: integer, *specific-trap-id*: integer, *agent-ip-address*: text, *agent-hostname*: text)

Argument	Description
<i>enterprise-id</i>	Defines a unique vendor-specific device. It is the standard way of identifying an organization or company. The <i>enterprise-id</i> is included in the event header defined in the SNMP protocol that is passed as a part of every SNMP event. It is a component of an SNMP Object Identifier (OID) in dot notation. For example, Gensym's enterprise ID is 1.3.6.1.4.1.1097.
<i>generic-trap-id</i>	Generic and specific trap IDs refer to two fields in an SNMP trap definition which, along with the enterprise ID, identify a trap event uniquely. The <i>generic-trap-id</i> field can contain values 0 - 6. Values 0 - 5 refer to generic events, such as a warm start or a cold start. A value of 6 indicates that this is an enterprise-specific trap and that the <i>specific-trap-id</i> value is meaningful (it is set to zero for generic traps).

Argument	Description
<i>specific-trap-id</i>	Specifies an enterprise-specific trap event (if <i>generic-trap-id</i> is set to 6). Set to 0 if <i>generic-trap-id</i> specifies a generic event (values 0 - 5).
<i>agent-ip-address</i>	Specify none . Not yet implemented.
<i>agent-hostname</i>	Specify none . Not yet implemented.

Enables a G2 knowledge base to add a trap defined by an enterprise id, generic trap id, and specific trap id to the list of traps that the SGB filters. If the specified trap is already in the list, it is not duplicated; the existing entry is used. Filtered traps are deleted at the SGB and not passed to G2.

For example, to ignore all traps with an enterprise id of "1.3.6.1.4.1.597.2.1", a generic id of 6 (meaning this is an enterprise-specific trap), and a specific id of 7, invoke `G2SNMP_ADD_FILTERED_TRAP()` with these parameters. At the time the remote procedure is invoked, the SGB begins deleting any traps matching this enterprise/generic/specific signature; thus, they are not returned to G2.

This RPC now supports the use of the agent hostname and agent address parameters with the following restrictions:

- Either an agent hostname or an agent IP address may be specified, but not both. If an agent hostname is specified, then the agent IP address must be set to the null string, "". If an agent IP address is specified, then the agent hostname must be set to the null string, "".
- If either an agent hostname or an agent IP address is specified, then the enterprise, generic, and specific parameters to this RPC must be set to the null string (""), zero (0), and zero (0), respectively.
- If the enterprise, generic, and specific parameters are specified, then the agent hostname and agent IP address both must be set to the null string, "".

The following table summarizes these restrictions. To use the table, find the parameter of interest in the leftmost column, then read across the table to determine what other parameters must be specified, and in appropriate cases, the values of those parameters.

Parameter of Interest	Agent Hostname	Agent Address	Enterprise ID	Generic ID	Specific ID
Agent hostname	Specified	"" (null string)	""(null string)	0	0
Agent address	""(null string)	Specified	""(null string)	0	0
Enterprise ID	""(null string)	""(null string)	Specified	Specified	Specified

Parameter of Interest	Agent Hostname	Agent Address	Enterprise ID	Generic ID	Specific ID
Generic ID	""(null string)	""(null string)	Specified	Specified	Specified
Specific ID	""(null string)	""(null string)	Specified	Specified	Specified

For G2SNMP_ADD_FILTERED_TRAP. If the agent hostname or IP address does not exist in the filtered hosts list residing in the G2-SNMP bridge, it is added and its filtering status is turned ON. If the specified agent already exists in the filtered hosts list, then its filtering status is turned ON. A filtering status of ON means that the current filtering behavior applies to this agent entry. That is, if the filtering mode is BLOCK, then traps from the specified agent are blocked. Likewise, if the filtering mode is PASS, then traps from the specified agent are passed.

The following table summarizes the effects of this RPC and its complementary RPC, G2SNMP_DELETE_FILTERED_TRAP. To use the table, find the RPC of interest in the leftmost column, then read across the table to determine the effect on filter status for the agent specified and the effect on filtering for different filtering modes.

RPC Applied	OFF (0)	PASS (1)	BLOCK (2)
g2snmp_add_filtered_trap	Filter Status ON Passed	Filter Status ON Passed	Filter Status ON Blocked
g2snmp_delete_filtered_trap	Filter Status OFF Passed	Filter Status OFF Blocked	Filter Status OFF Passed

g2snmp_delete_filtered_trap**g2snmp_delete_filtered_trap**

(*enterprise-id*: text, *generic-trap-id*: integer, *specific-trap-id*: integer, *agent-IP-address*: text, *agent-hostname*: text)

Argument	Description
<i>enterprise-id</i>	Defines a unique vendor-specific device. It is the standard way of identifying an organization or company. The <i>enterprise-id</i> is included in the event header defined in the SNMP protocol that is passed as a part of every SNMP event. It is a component of an SNMP Object Identifier (OID) in dot notation. For example, Gensym's enterprise ID is 1.3.6.1.4.1.1097.
<i>generic-trap-id</i>	Generic and specific trap IDs refer to two fields in an SNMP trap definition which, along with the enterprise ID, identify a trap event uniquely. The <i>generic-trap-id</i> field can contain values 0 - 6. Values 0 - 5 refer to generic events, such as a warm start or a cold start. A value of 6 indicates that this is an enterprise-specific trap and that the <i>specific-trap-id</i> value is meaningful (it is set to zero for generic traps).
<i>specific-trap-id</i>	Specifies an enterprise-specific trap event (if <i>generic-trap-id</i> is set to 6). Set to 0 if <i>generic-trap-id</i> specifies a generic event (values 0 - 5).
<i>agent-IP-address</i>	Specify none . Not yet implemented.
<i>agent-hostname</i>	Specify none . Not yet implemented.

Enables a G2 knowledge base to disable filtering a trap defined by an enterprise id, generic trap id, and specific trap id from the list of traps that will be filtered by the SGB. The trap is not removed from the list and filtering may again be enabled by using `G2SNMP_ADD_FILTERED_TRAP()`. Filtered traps are deleted at the SGB and not passed to G2.

For example, to begin receiving a previously filtered trap with an enterprise id of "1.3.6.1.4.1.597.2.1", a generic id of 6 (meaning this is an enterprise-specific trap),

and a specific id of 7, invoke `G2SNMP_DELETE_FILTERED_TRAP()` with these parameters. At the time the remote procedure is invoked, the SGB stops deleting traps matching this enterprise/generic/specific signature, and returns them to G2.

This RPC now supports the use of the agent hostname and agent address parameters with the following restrictions:

- Either an agent hostname or an agent IP address may be specified, but not both. If an agent hostname is specified, then the agent IP address must be set to the null string, "". If an agent IP address is specified, then the agent hostname must be set to the null string, "".
- If either an agent hostname or an agent IP address is specified, then the enterprise, generic, and specific parameters to this RPC must be set to the null string (""), zero (0), and zero (0), respectively.
- If the enterprise, generic, and specific parameters are specified, then the agent hostname and agent IP address both must be set to the null string, "".

The following table summarizes these restrictions. To use the table, find the parameter of interest in the leftmost column, then read across the table to determine what other parameters must be specified, and in appropriate cases, the values of those parameters.

Parameter of Interest	Agent Hostname	Agent Address	Enterprise ID	Generic ID	Specific ID
Agent hostname	Specified	"" (null string)	""(null string)	0	0
Agent address	""(null string)	Specified	""(null string)	0	0
Enterprise ID	""(null string)	""(null string)	Specified	Specified	Specified
Generic ID	""(null string)	""(null string)	Specified	Specified	Specified
Specific ID	""(null string)	""(null string)	Specified	Specified	Specified

The `G2SNMP_DELETE_FILTERED_TRAP` RPC may be used only on agents that already have an entry in the filtered hosts list residing within the G2-SNMP bridge. This RPC turns the filtering status for the specified agent to OFF. A filtering status of OFF means that the current filtering behavior does not apply to this agent entry. That is, if the filtering mode is BLOCK, then traps from the specified agent are passed. Likewise, if the filtering mode is PASS, then traps from the specified agent are blocked. The filtering mode for the specified agent may be toggled back to ON using the `G2SNMP_ADD_FILTERED_TRAP` RPC.

g2snmp_modify_comm_params

g2snmp_modify_comm_params

*(time-out-interval: integer, retry-count: integer)**-> error-code: integer, error-string: text*

Argument	Description
<i>time-out-interval</i>	The length of time, in tenths of a second, before the SGB attempts to try sending the request again.
<i>retry-count</i>	The number of times the SGB makes a request after the initial request. For example, if <i>retry-count</i> = 3 and the SGB sends an initial request that times out, the SGB sends three additional identical requests before returning an error. The <i>error-string</i> is: "Error sending request: No response arrived before timeout."
Return Value	Description
<i>error-code</i>	Returns a value of 0 if the transaction is successful else the HP OpenView or NetView 6000 integer error code.
<i>error-string</i>	Returns a value of "[the current value of the time-out-interval]-[the current value of the retry-count]" if the transaction is successful else "[the text translation of the error code as defined in the HP OpenView or NetView 6000 documentation]"

Enables a G2 knowledge base to modify the request time-out interval and the number of retries for sending the request.

g2snmp_set_agent_filter_mode

This RPC accommodates setting the agent hostname/IP address filtering mode to one of three permissible values: 0 (OFF-no agent hostname/IP address filtering); 1 (PASS-allow only traps from the specified agent hostnames/IP addresses to pass); and 2 (BLOCK-block all traps from the specified agent hostnames/IP addresses). See the following section, "Update for GSI-based Bridge Process," for details on the effects of these various modes.

This G2-based RPC activates the G2SNMP_setAgentFilterMode() RPC in the GSI-based portion of the bridge. It accepts the following arguments: integer (filter_mode). In a G2 application, add the following RPC definition:

```
g2snmp_set_agent_filter_mode(integer; { Filter mode: 0, 1, 2 }) =
    (integer, text)
```

Set the 'Name in remote system' entry in the remote procedure declaration table to 'G2SNMP_SET_AGENT_FILTER_MODE.' This procedure can be either 'start'-ed or 'call'-ed from the G2 application. If called, it returns an integer status code and a status message string.

g2snmp_use_snmp_comm_params

```
g2snmp_use_snmp_comm_params
    ()
    -> error-code: integer, error-string: text
```

Return Value	Description
<i>error-code</i>	Returns a value of 0.
<i>error-string</i>	Returns a value of GSI_SUCCESS.

Enables a G2 knowledge base to configure the SGB to use the user-specified values for the request time-out interval (length of time before the SGB attempts to try sending the request again in tenths of a second) and the number of retries (after the initial request) for re-sending the request. These values are set using g2snmp_modify_comm_params.

Note When the SGB is configured to use the user-specified parameters through G2SNMP_USE_SNMP_COMM_PARAMS(), the retry and timeout values are applied to *all* subsequent non-blocking SNMP GETs and SETs for *all* agents.

g2snmp_use_snmp_defaults

```
g2snmp_use_snmp_defaults
    ()
    -> error-code: integer, error-string: text
```

Return Value	Description
<i>error-code</i>	Returns a value of 0.
<i>error-string</i>	Returns a value of GSI_SUCCESS.

Enables a G2 knowledge base to configure the SGB to use the default values for the request time-out interval (length of time before the SGB attempts to try

sending the request again in tenths of a second) and the number of retries (after the initial request) for re-sending the request. These parameters apply to all non-blocking SNMP GETs and SETs. The default parameter values are contained in the *ovsnmp.conf* file. See the HP OpenView documentation for configuring the defaults in this file.

Note The currently supported version of HP OpenView allows configuring different retry and timeout values for each SNMP agent.

g2snmp_blocking_transaction

g2snmp_blocking_transaction

(*transaction-tagname*: text, *request-code*: integer, *node-name*: text, *community-name*: text, *variable-oid*: text, *variable-ASN1-type*: integer, *variable-value*: text)
 -> *error-code*: integer, *error-string*: text, *node-name*: text

Argument	Description
<i>transaction-tagname</i>	A handle for matching an incoming response with the outgoing request.
<i>request-code</i>	The type of blocking transaction requested. Possible values are: 160 (GET request) 161 (GETNEXT request) 162 (GETRESPONSE request) 163 (SET request)
<i>node-name</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	A string specifying the administrative relationship for the transaction.
<i>variable-oid</i>	The object identifier (<i>OID</i>), usually in dotted notation, for the variable involved in the GET or SET transaction.

Argument	Description
<i>variable-ASN1-type</i>	<p>The ASN.1 type of the variable involved in the GET or SET transaction. ASN.1 is a formal language for describing data and the properties of data.</p> <p><i>variable-ASN1-type</i> is set to one of the following integer constants:</p> <p style="padding-left: 40px;">2 (integer)</p> <p style="padding-left: 40px;">4 (octet string)</p>
<i>variable-value</i>	The new value of the variable in a SET transaction.

Return Value	Description
<u><i>error-code</i></u>	Returns a value of 0 if the transaction completes successfully else the HP OpenView or NetView 6000 integer error code.
<u><i>error-string</i></u>	Returns a value of NO ERRORS if the transaction completes successfully else “[the text translation of the error code as defined in the HP OpenView or NetView 6000 documentation]”.
<u><i>node-name</i></u>	Returns the node name to which the transaction was directed.

Enables a G2 knowledge base to perform a blocking SNMP transaction.

g2snmp_nonblocking_transaction**g2snmp_nonblocking_transaction**

(*transaction-tagname*: text, *request-code*: integer, *node-name*: text, *community-name*: text, *variable-oid*: text, *variable-ASN1-type*: integer, *variable-value*: text)
 -> *result-id*: integer

Argument	Description
<i>transaction-tagname</i>	A handle for identifying the outgoing request.
<i>request-code</i>	The type of non-blocking transaction being requested. Possible values: 160 (GET request) 164 (TRAP send request)
<i>node-name</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>variable-oid</i>	The object identifier (OID), usually in dotted notation, for the variable involved in the GET or TRAP send transaction.
<i>variable-ASN1-type</i>	The ASN.1 type of the variable involved in the GET or TRAP send transaction. Possible values are: 4 (octet string) 2 (integer)
<i>variable-value</i>	The variable value is the value of the variable in a TRAP send transaction.

Return Value	Description
<u><i>result-id</i></u>	Returns a handle to the MIB-RECEIVER object that will receive the results of the SNMP GET transaction. This value will match the value of the Result-id of the MIB-RECEIVER object.

Enables a G2 knowledge base to perform a non-blocking SNMP transaction (a GET or a TRAP send).

Overloaded RPCs

The following remote procedure calls are overloaded RPC's for use in performing various GET, SET, and trap SEND SNMP transactions.

get_nonblocking_single

get_nonblocking_single

(*transaction-tagname*: text, *request-code*: integer, *machine-nodename*: text, *community-name*: text, *OID*: text)
-> *result-id*: integer

Argument	Description
<i>transaction-tagname</i>	A handle for identifying the outgoing request.
<i>request-code</i>	The type of non-blocking transaction being requested. Possible values are: snmp_get_req_msg = 160 (GET Request) snmp_getnext_req_msg = 161 (GET NEXT Request)
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>OID</i>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.

Return Value	Description
<u>result-id</u>	Returns a handle to the MIB-RECEIVER object that will receive the results of the SNMP GET transaction. This value will match the value of the Result-id of the MIB-RECEIVER object.

Performs a single non-blocking GET transaction and returns one value the *result-id*. The *result-id* is the unique identifier of the MIB-RECEIVER object where the result of the GET transaction is stored.

get_blocking_single

get_blocking_single

(*transaction-tagname*: text, *request-code*: integer, *machine-nodename*: text, *community-name*: text, *OID*: text)

-> *error-code*: integer, *error-string*: text, *machine-nodename*: text, *text-OID*: text, *text-value*: text, *integer-OID*: text, *integer-value*: integer, *float-OID*: text, *float-value*: float

Argument	Description
<i>transaction-tagname</i>	A handle for indentifying the outgoing request.
<i>request-code</i>	The type of blocking transaction being requested. Possible values are: snmp_get_req_msg = 160 (GET Request) snmp_getnext_req_msg = 161 (GET NEXT Request)
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>OID</i>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.

Return Value	Description
<u>error-code</u>	Returns a value of 0 if the transaction is successful.
<u>error-string</u>	Returns a value of NO ERRORS if the transaction is successful.
<u>machine-nodename</u>	The nodename or IP address of the machine to which the GET transaction was directed.
<u>text-OID</u>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
<u>text-value</u>	The variable requested by the GET transaction for <i>OID</i> .
<u>integer-OID</u>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
<u>integer-value</u>	The variable requested by the GET transaction for <i>OID</i> .
<u>float-OID</u>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
<u>float-value</u>	The variable requested by the GET transaction for <i>OID</i> .

Performs a single blocking SNMP GET transaction and returns the results as part of the call. Depending upon the type of the value being returned, the return value is placed in one of the following fields: text-value, integer-value or float-value,

get_2_blocking

Performs a blocking SNMP GET transaction on two variables. The results are returned as part of the call.

get_2_blocking

(*transaction-tagname*: text, *request-code*: integer, *machine-nodename*: text, *community-name*: text, *OID-1*: text, *OID-2*: text)

-> (*error-code*: integer, *error-string*: text, *machine-nodename*: text, *OID-1*: text, *value-1*: text, *OID-2*: text, *value-2*: integer)

Argument	Description
<i>transaction-tagname</i>	A handle for indentifying the outgoing request.
<i>request-code</i>	The type of blocking transaction being requested. Possible values are: snmp_get_req_msg = 160 (GET Request) snmp_getnext_req_msg = 161 (GET NEXT Request)
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>OID-1</i>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
<i>OID-2</i>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
Return Value	Description
<u><i>error-code</i></u>	Returns a value of 0 if the transaction is successful.
<u><i>error-string</i></u>	Returns a value of NO ERRORS if the transaction is successful.
<u><i>machine-nodename</i></u>	The nodename or IP address of the machine to which the GET transaction was directed.
<u><i>OID-1</i></u>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
<u><i>value-1</i></u>	The text variable requested by the GET transaction for <i>OID-1</i> .

Return Value	Description
<u>OID-2</u>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
<u>value-2</u>	The integer variable requested by the GET transaction for <i>OID-2</i> .

set_blocking

Use this procedure to perform an SNMP SET transaction.

set_blocking

(*transaction-tagname*: text, *request-code*: integer, *machine-nodename*: text, *community-name*: text, *OID*: text, *ASN1-Type*: integer, *new-value*: text)
 -> (*error-code*: integer, *error-string*: text, *machine-nodename*: text)

Argument	Description
<i>transaction-tagname</i>	A handle for indentifying the outgoing request.
<i>request-code</i>	The type of blocking transaction being requested. snmp_set_req_msg = 163 (SET Request)
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>OID</i>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
<i>ASN1-Type</i>	The ASN.1 type of the variable involved in the SET transaction. Possible values are: asn1_integer (2) asn1_octet_string (4)
<i>new-value</i>	The new value of the <i>OID</i> , as a text string.

Return Value	Description
<i><u>error-code</u></i>	Returns a value of 0 if the transaction is successful else returns the HP OpenView or NetView 6000 integer error code.
<i><u>error-string</u></i>	Returns a value of NO ERRORS if the transaction is successful else “[the text translation of the error code as defined in the HP OpenView or NetView 600 documentation]”.
<i><u>machine-nodename</u></i>	The nodename or IP address of the machine to which the SET transaction was directed.

set_nonblocking_integer

Use this procedure to perform an SNMP SET transaction for an integer value.

set_nonblocking_integer

(*transaction-tagname*: text, *request-code*: integer, *machine-nodename*: text, *community-name*: text, *OID*: text, *ASN1-Type*: integer, *new-value*: integer)
 -> (*result-id*: integer)

Argument	Description
<i>transaction-tagname</i>	A handle for indentifying the outgoing request.
<i>request-code</i>	The type of blocking transaction being requested. snmp_set_req_msg = 163 (SET Request)
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>OID</i>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.

Argument	Description
<i>ASN1-Type</i>	The ASN.1 type of the variable involved in the SET transaction. Possible values are: asn1_integer (2)
<i>new-value</i>	The new value of the <i>OID</i> , as an integer.

Return Value	Description
<u><i>result-id</i></u>	Returns a value of -1 if no SNMP sessions could be opened in the G2-SNMP bridge, else a positive number used for a handle to the result.

set_nonblocking_text

Use this procedure to perform an SNMP SET transaction for a text value.

set_nonblocking_text

(*transaction-tagname*: text, *request-code*: integer, *machine-nodename*: text, *community-name*: text, *OID*: text, *ASN1-Type*: integer, *new-value*: text)
-> (*result-id*: integer)

Argument	Description
<i>transaction-tagname</i>	A handle for indentifying the outgoing request.
<i>request-code</i>	The type of blocking transaction being requested. snmp_set_req_msg = 163 (SET Request)
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>OID</i>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.

Argument	Description
<i>ASN1-Type</i>	The ASN.1 type of the variable involved in the SET transaction. Possible values are: asn1_octet_string (4)
<i>new-value</i>	The new value of the <i>OID</i> , as a string.
Return Value	Description
<u><i>result-id</i></u>	Returns a value of -1 if no SNMP sessions could be opened in the G2-SNMP bridge, else a positive number used for a handle to the result.

send_novar_trap_nonblocking

Use this procedure to send a non-blocking SNMP trap with no values.

send_novar_trap_nonblocking

(*transaction-tagname*: text, *request-code*: integer, *machine-nodename*: text, *community-name*: text, *enterprise-id*: text, *agent-ip-address*: text, *generic-trap-id*: integer, *specific-trap-id*: integer, *agent-run-time*: integer)

Argument	Description
<i>transaction-tagname</i>	A handle for indentifying the outgoing request.
<i>request-code</i>	The type of blocking transaction being requested. snmp_trap_req_msg = 164 (TRAP Send Request)
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.

Argument	Description
<i>enterprise-id</i>	<p>Defines a unique vendor-specific device. It is the standard way of identifying an organization or company.</p> <p>The <i>enterprise-id</i> is included in the event header defined in the SNMP protocol that is passed as a part of every SNMP event. It is a component of an SNMP Object Identifier (OID) in dot notation.</p> <p>For example, Gensym's enterprise ID is 1.3.6.1.4.1.1097.</p>
<i>agent-IP-address</i>	<p>The name or dotted IP address of the agent node to which the transaction is directed.</p>
<i>generic-trap-id</i>	<p>Generic and specific trap IDs refer to two fields in an SNMP trap definition which, along with the enterprise ID, identify a trap event uniquely.</p> <p>The <i>generic-trap-id</i> field can contain values 0 - 6. Values 0 - 5 refer to generic events, such as a warm start or a cold start. A value of 6 indicates that this is an enterprise-specific trap and that the <i>specific-trap-id</i> value is meaningful (it is set to zero for generic traps).</p>
<i>specific-trap-id</i>	<p>Specifies an enterprise-specific trap event (if <i>generic-trap-id</i> is set to 6).</p> <p>Set to 0 if <i>generic-trap-id</i> specifies a generic event (values 0 - 5).</p>
<i>agent-run-time</i>	<p>The amount of time elapsed between the last initialization of the agent and the generation of the trap.</p>

send_trap_nonblocking

Use this procedure to send a non-blocking SNMP trap with one value.

send_trap_nonblocking

(*transaction-tagname*: **text**, *request-code*: **integer**, *machine-nodename*: **text**, *community-name*: **text**, *enterprise-id*: **text**, *agent-ip-address*: **text**, *generic-trap-id*: **integer**, *specific-trap-id*: **integer**, *agent-run-time*: **integer**, *OID*: **text**, *ASN1-Type*: **integer**, *trap-value*: **text**)

Argument	Description
<i>transaction-tagname</i>	A handle for indentifying the outgoing request.
<i>request-code</i>	The type of blocking transaction being requested. snmp_trap_req_msg = 164 (TRAP Send Request)
<i>machine-nodename</i>	The name or dotted IP address of the manager node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>enterprise-id</i>	Defines a unique vendor-specific device. It is the standard way of identifying an organization or company. The <i>enterprise-id</i> is included in the event header defined in the SNMP protocol that is passed as a part of every SNMP event. It is a component of an SNMP Object Identifier (OID) in dot notation. For example, Gensym's enterprise ID is 1.3.6.1.4.1.1097.
<i>agent-IP-address</i>	The name or dotted IP address of the agent node to which the transaction is directed.

Argument	Description
<i>generic-trap-id</i>	<p>Generic and specific trap IDs refer to two fields in an SNMP trap definition which, along with the enterprise ID, identify a trap event uniquely.</p> <p>The <i>generic-trap-id</i> field can contain values 0 - 6. Values 0 - 5 refer to generic events, such as a warm start or a cold start. A value of 6 indicates that this is an enterprise-specific trap and that the <i>specific-trap-id</i> value is meaningful (it is set to zero for generic traps).</p>
<i>specific-trap-id</i>	<p>Specifies an enterprise-specific trap event (if <i>generic-trap-id</i> is set to 6). Set to 0 if <i>generic-trap-id</i> specifies a generic event (values 0 - 5).</p>
<i>agent-run-time</i>	<p>The amount of time elapsed between the last initialization of the agent and the generation of the trap.</p>
<i>OID</i>	<p>The object identifier, usually in dotted notation, for the variable involved in the TRAP send transaction.</p>
<i>ASN1-Type</i>	<p>The ASN.1 type of the variable involved in the SET transaction. Possible values are:</p> <p>asn1_integer (2)</p> <p>asn1_octet_string (4)</p>
<i>trap-value</i>	<p>The new value of the <i>OID</i>.</p>

send_trap_status_nonblocking

Use this procedure to send a non-blocking SNMP trap with four values.

send_trap_status_nonblocking

(*transaction-tagname*: text, *request-code*: integer, *machine-nodename*: text, *community-name*: text, *enterprise-id*: text, *agent-ip-address*: text, *generic-trap-id*: integer, *specific-trap-id*: integer, *agent-run-time*: integer, *OID-1*: text, *ASN1-Type-1*: integer, *trap-value-1*: integer, *OID-2*: text, *ASN1-Type-2*: integer, *trap-value-2*: text, *OID-3*: text, *ASN1-Type-3*: integer, *trap-value-3*: text, *OID-4*: text, *ASN1-Type-4*: integer, *trap-value-4*: text)

Argument	Description
<i>transaction-tagname</i>	A handle for indentifying the outgoing request.
<i>request-code</i>	The type of blocking transaction being requested. snmp_trap_req_msg = 164 (TRAP Send Request)
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>enterprise-id</i>	Defines a unique vendor-specific device. It is the standard way of identifying an organization or company. The <i>enterprise-id</i> is included in the event header defined in the SNMP protocol that is passed as a part of every SNMP event. It is a component of an SNMP Object Identifier (OID) in dot notation. For example, Gensym's enterprise ID is 1.3.6.1.4.1.1097.
<i>agent-IP-address</i>	The name or dotted IP address of the agent node to which the transaction is directed.

Argument	Description
<i>generic-trap-id</i>	<p>Generic and specific trap IDs refer to two fields in an SNMP trap definition which, along with the enterprise ID, identify a trap event uniquely.</p> <p>The <i>generic-trap-id</i> field can contain values 0 - 6. Values 0 - 5 refer to generic events, such as a warm start or a cold start. A value of 6 indicates that this is an enterprise-specific trap and that the <i>specific-trap-id</i> value is meaningful (it is set to zero for generic traps).</p>
<i>specific-trap-id</i>	<p>Specifies an enterprise-specific trap event, if the <i>generic-trap-id</i> is set to 6.</p> <p>Set to 0 if <i>generic-trap-id</i> specifies a generic event (values 0 - 5).</p>
<i>agent-run-time</i>	The amount of time elapsed between the last (re)-initialization of the agent and the generation of the trap.
<i>OID-1</i>	The object identifier, usually in dotted notation, for the variable involved in the TRAP send transaction.
<i>ASN1-Type-1</i>	<p>The ASN.1 type of the variable involved in the SET transaction. Possible values are:</p> <p><code>asn1_integer (2)</code></p> <p><code>asn1_octet_string (4)</code></p>
<i>trap-value-1</i>	The new value of the <i>OID</i> .
<i>OID-2</i>	The object identifier, usually in dotted notation, for the variable involved in the TRAP send transaction.
<i>ASN1-Type-2</i>	<p>The ASN.1 type of the variable involved in the SET transaction. Possible values are:</p> <p><code>asn1_integer (2)</code></p> <p><code>asn1_octet_string (4)</code></p>
<i>trap-value-2</i>	The new value of the <i>OID</i> .

Argument	Description
<i>OID-3</i>	The object identifier, usually in dotted notation, for the variable involved in the TRAP send transaction.
<i>ASN1-Type-3</i>	The ASN.1 type of the variable involved in the SET transaction. Possible values are: asn1_integer (2) asn1_octet_string (4)
<i>trap-value-3</i>	The new value of the <i>OID</i> .
<i>OID-4</i>	The object identifier, usually in dotted notation, for the variable involved in the TRAP send transaction.
<i>ASN1-Type-4</i>	The ASN.1 type of the variable involved in the SET transaction. Possible values are: asn1_integer (2) asn1_octet_string (4)
<i>trap-value-4</i>	The new value of the <i>OID</i> .

Receiver Procedures

G2 procedures that the SNMP Gateway Bridge calls to return data to G2. The SGB can call these functions to return:

- Data requested by non-blocking requests.
- Error messages.
- Traps received by the SGB through the SNMP trap receiver process.

g2snmp_receive_eot

Returns an end-of-transaction acknowledgment message to G2 from the SGB for non-blocking transactions (such as an SNMP trap send or a non-blocking SNMP get) . The message is formatted as:

"In [the current procedure name] RECEIVED end-of-transaction for
Transaction type: [transaction_type text] RESULT-ID: [result_id value]
number of variables: [counter value] context: [context value]".

The level of the *mib-debug* parameter determines when messages are sent to the *devu-error-handler()* for display, either on the message board or by other methods that the you have specified.

g2snmp_receive_eot

(*transaction_type*: text, *result_id*: integer, *counter*: integer, *context*: integer)

Argument	Description
<i>transaction_type</i>	Either TRAP or NON-BLOCKING.
<i>result_id</i>	The return code generated by the transaction request.
<i>counter</i>	Number of variables transferred as part of the transaction request.
<i>context</i>	The SGB context in which the transaction has been processed. Since multiple G2 processes may connect to a single SGB, and multiple Telewindows sessions may be logged in to any of the G2 processes, the context value helps the user track the transactions of interest.

g2snmp_receive_float

Receives a float in G2 from the SNMP Gateway Bridge.

g2snmp_receive_float

(*error_code*: integer, *error_string*: text, *result_id*: integer, *node-name*: text, *object-id*: text, *result*: float)

Argument	Description
<i>error_code</i>	Request status. A return value of zero (0) indicates success; otherwise, an error condition occurred.
<i>error_string</i>	Text description of the request status. A returned value of NO ERRORS indicates success; otherwise, an error condition occurred.
<i>result_id</i>	This integer is used to correlate values which are returned by the SGB to the request made by the non-blocking call or trap.
<i>node-name</i>	The <i>node-name</i> from which the variables are requested. This is returned as either the hostname or the ip-address.

Argument	Description
<i>object-id</i>	Object-identifier of the returned variable.
<i>result</i>	The value of the requested float variable.

g2snmp_receive_integer

Receives an integer in G2 from the SNMP Gateway Bridge.

g2snmp_receive_integer

(*error_code*: integer, *error_string*: text, *result_id*: integer, *node-name*: text, *object-id*: text, *result*: integer)

Argument	Description
<i>error_code</i>	Request status. A return value of zero (0) indicates success; otherwise, an error condition occurred.
<i>error_string</i>	Text description of the request status. A returned value of NO ERRORS indicates success; otherwise, an error condition occurred.
<i>result_id</i>	This integer is used to correlate values which are returned by the SGB to the request made by the non-blocking call or trap.
<i>node-name</i>	The <i>node-name</i> from which the variables are requested. This is returned as either the hostname or the ip-address.
<i>object-id</i>	Object-identifier of the returned variable.
<i>result</i>	The value of the requested integer variable.

g2snmp_receive_message

Receives a message in G2 from the SNMP Gateway Bridge. The message is formatted as:

"In [*the current procedure name*] tag: [*tag text*] node: [*node text*] error_string: [*error_string text*] error_code: [*error_code value*]".

The level of the `mib-debug` parameter determines when messages are sent to the `devu-error-handler()` for display, either on the message board or by other methods that the you have specified.

`g2snmp_receive_message` is intended to be used as a general-purpose means of sending a message back to G2. No verification of any of the returned fields is performed. It is assumed that the SGB has filled in the values properly.

`g2snmp_receive_message`

(*tag*: text, *node*: text, *error_string*: text, *error_code*: integer)

Argument	Description
<i>tag</i>	General text to be displayed as a preface to the message.
<i>node</i>	The <i>node-name</i> from which the message was sent. This may be returned as either the hostname or the ip-address.
<i>error_string</i>	Text description of the message status.
<i>error_code</i>	Message status.

`g2snmp_receive_string`

Receives a string in G2 from the SNMP Gateway Bridge.

`g2snmp_receive_string`

(*error_code*: integer, *error_string*: text, *result_id*: integer, *node-name*: text, *object-id*: text, *result*: text)

Argument	Description
<i>error_code</i>	Request status. A return value of zero (0) indicates success; otherwise, an error condition occurred.
<i>error_string</i>	Text description of the request status. A returned value of NO ERRORS indicates success; otherwise, an error condition occurred.
<i>result_id</i>	This integer is used to correlate values which are returned by the SGB to the request made by the non-blocking call or trap.
<i>node-name</i>	The <i>node-name</i> from which the variables are requested. This is returned as either the hostname or the ip-address.

Argument	Description
<i>object-id</i>	Object-identifier of the returned variable.
<i>result</i>	The value of the requested string variable.

g2snmp_receive_trap_packet

Receives a single packet of trap packet information in G2 from the SNMP Gateway Bridge. The trap receive mechanism differs significantly from the other receiver functions. In order to improve performance in handling large numbers of incoming traps (for example, during a trap 'storm'), the values contained in the trap protocol data unit (PDU) are not returned to G2 individually. Instead, the PDU information is packetized as a standard header containing PDU and packet information, plus a packet body of up to sixteen PDU *oid/value* pairs per packet. The structure of the packet parallels the order of the arguments to `g2snmp_receive_trap_packet`. The header consists of *error_code*, *error_string*, *result_id*, *enterprise_id*, *agent_address*, *agent_hostname*, *generic_trap*, *specific_trap*, *agent_act_time*, *seq_number*, *number_packet*, *vars_in_packet*.

The packet body consists of *oid/value* pairs. Each of these values are described below. If there are more than sixteen *oid/value* pairs to be returned to G2 from the trap, they are split across multiple packets, each with a similar header (*error_code* through *agent_act_time* and *number_packet* the same in each packet, *seq_number* and *vars_in_packet* varying to reflect the individual packet).

Although the SGB ships the packets to G2 in the proper order, the SGB cannot guarantee in-order delivery to G2. Therefore, any processing done by G2 on the received packet should use the *seq_number* field to identify the group of *oid/value* pairs returned in the packet.

g2snmp_receive_trap_packet

(*error_code*: integer, *error_string*: text, *result_id*: integer, *enterprise_id*: text, *agent_address*: text, *agent_hostname*: text, *generic_trap*: integer, *specific_trap*: text, *agent_act_time*: integer, *seq_number*: integer, *number_packet*: integer, *vars_in_packet*: integer *oid_1*: text, *val_1*: text, *oid_N*: text, *val_N*: text)

Argument	Description
<i>error_code</i>	Request status. A return value of zero (0) indicates success; otherwise, an error condition occurred.
<i>error_string</i>	Text description of the request status. A returned value of "NO ERRORS" indicates success; otherwise, an error condition occurred.

Argument	Description
<i>result_id</i>	This integer is used to correlate values which are returned by the bridge to the request made by the non-blocking call or trap.
<i>enterprise_id</i>	A dot-notation oid defining the manufacturer of the device sending the trap.
<i>agent_address</i>	The dot-notation IP address of the agent sending the trap.
<i>agent_hostname</i>	The name of the host on which the agent sending the trap resides.
<i>generic_trap</i>	SNMP generic trap code.
<i>specific_trap</i>	Enterprise-specific trap code.
<i>agent_act_time</i>	Length of time the agent sending the trap has been active.
<i>seq_number</i>	Sequence number of the packet. If more than sixteen oid/value pairs will be returned to G2, <i>seq_number</i> will be incremented by the bridge for each packet sent.
<i>number_packet</i>	Total number of packets that this trap comprises.
<i>vars_in_packet</i>	Number of oid/value pairs in this packet. May be zero (0).
<i>oid_1, val_1</i>	Object-identifier and value of the first returned variable. If the <i>vars_in_packet</i> parameter is zero (0), no oid/value pairs are present.
<i>oid_N, val_N</i>	Object-identifier and value of the last returned variable. Present if more than one (1) oid/value pair is returned. Up to sixteen (16) oid/value pairs may be returned in a single packet.

Two arguments, `community-name` and `community-length`, have been added to the argument lists of this RPC. The remainder of the argument lists remains the same. The argument lists now have the form:

```
g2snmp_receive_trap_packet
(
  error-code: integer, error-string: text, result-id: integer, enterprise-id: text,
  agent-address: text, agent-hostname: text, generic-trap: integer, specific-trap: text,
  agent-act-time: integer, community-name: text, community-length: integer ...
)
```

Procedures Listed by Module

The API procedures are listed according to the module that contains them. The modules that include API procedures are as follows:

- GNDO Module - Developer utility module which contains procedures for parsing and looking up domain objects.
- GMIB Module - Management Information Base module which contains procedures for reading ASN.1 MIB's and the *trapd.conf* file.
- GSNMP Module - Integrity SNMP simulation module, which contains procedures for the simulation of SNMP traps, objects for a simulated agent MIB, and procedures for performing various SNMP transactions.

GNDO Module

The GNDO module contains procedures for parsing and domain object lookup.

devu-consume-next-field

(*txt*: text, *separator*: text)

-> (*next-field*: text, *text-remaining*: text)

Parses a text string containing fields separated by a unique separator text. Returns two text strings. *next-field* contains *txt* up to, but not including, the first occurrence of *separator*. *text-remaining* contains what is left in *txt* after *next-field* and the first separator are stripped off.

devu-decode-comma-line

(*text-string*: text, *txt-list*: class text-list)

Use this procedure to turn a comma separated text string into a list of text items. Breaks comma separated strings in *text-string* into individual strings and inserts them into the list *txt-list* without any duplicate items.

devu-domain-object-lookup

(*object-name*: text)

-> (*domain-object*: class [opfo-managed-object | opfo-containment-object])

Returns the *domain-object* with the specified *object-name* as the value of its `_opfo-external-name` attribute or as its name. If you want to lookup an object based on a different object attribute, you can define a custom procedure which will be called by `devu-domain-object-lookup`. The name of this procedure is defined in the G2 parameter `devu-alternate-object-lookup-procedure`. The procedure name is initialized in this parameter using an initialization item.

If no object matching these criteria is found, by default, `devu-domain-object-lookup` returns the domain object `devu-safe-object`. If you want to return some other default object instead, you can define a custom procedure and specify the name of the procedure using the initialization for the parameter `devu-unknown-object-procedure`.

devu-error-handler

(*target*: class item, *sender*: class item, *error-category*: text, *priority*: integer, *errname*: symbol, *errtext*: text, *lifetime*:integer)

This is the default error handler in Integrity. This procedure executes the error handler procedure specified in the G2 parameter *devu-error-handler-proc*. You can define a custom error handler procedure and specify the name of the procedure using the initialization for *devu-error-handler-proc*.

devu-get-field-n

(*txt*: text, *n*: integer, *separator*: text)
-> (*nth-field*: text)

The argument *txt* is a text string containing fields separated by a *separator* text. This procedure returns the *nth-field* in *txt*. If *txt* does not have *n* fields, an empty string is returned.

devu-insert-item-in-sorted-ascending-list

(*new-item*: class item, *list*: class item-list, *key-proc*: class procedure)

Inserts *new-item* in *list* in ascending order based on criteria defined in *key-proc*. The procedure defined by *key-proc* should accept an *item* as an argument and return an integer which is used for sorting the items.

devu-insert-item-in-sorted-descending-list

(*new-item*: class item, *list*: class item-list, *key-proc*: class procedure)

Inserts *new-item* in *list* in descending order based on criteria defined in *key-proc*. The procedure defined by *key-proc* should accept an *item* as an argument and return an integer which is used for sorting the items.

devu-insert-msg-in-alphabetical-list

(*msg*: class message, *list*: class item-list)

Inserts a *msg* in alphabetical order into *list*. No duplicates are allowed in the list.

devu-insert-msg-in-alphabetical-list-allowing-duplicates

(*msg*: class message, *list*: class item-list)

Inserts a *msg* in alphabetical order into *list*. Duplicates are allowed in the list.

devu-pattern-matcher-with-indices

(*txt*: text, *txt-pos*: integer, *pattern*: text, *pattern-pos*: integer)
-> (*match-value*: truth-value)

Searches for *pattern* in *txt*. The symbol "*" in *pattern* is matched by any number of arbitrary characters. The symbol "?" in *pattern* is matched by exactly one arbitrary character. The characters "*?" together are not matched. The match begins starting from the position *pattern-pos* in the pattern and *txt-pos* in the text string *txt*. The procedure returns true if a match is found, otherwise false.

devu-safe-text-for-symbol

(*txt*: text)
 -> (*txt-to-symbol*: text)

Converts a text string, *txt*, to a valid G2 symbol string, *txt-to-symbol*. It replaces all spaces with "_", and prefixes special characters with "@". Spaces at the end of the string are removed.

devu-safe-window

()
 -> (*safe-window*: class g2-window)

Returns a safe G2 window.

devu-strip-linefeeds

(*txt*: text)
 -> (*txt-without-linefeeds*: text)

Strips the linefeeds from *txt* and returns the text without any linefeeds in *txt-without-linefeeds*.

devu-substitute-text

(*match-string*: text, *replace-string*: text, *txt*: text)
 -> (*new-txt*: text)

Finds every occurrence of *match-string* in *txt* and replaces it with *replace-string*. The new text string is returned in *new-txt*.

devu-txt-is-ascii-punctuation-p

(*character*: text)
 -> (*is-ASCII-character*: truth-value)

Returns true if *character* is an ASCII symbol, otherwise returns false.

The following value for *character*, enclosed in parenthesis, will return true:

" ", "!", "?", "@", "<", ";", "(", ")", ":", ";", or "."

devu-txt-is-ascii-symbol-p

(*character*: text)
 -> (*is-ASCII-punctuation*: truth-value)

Returns true if *character* is an ASCII punctuation, otherwise returns false.

The following value for *character*, enclosed in parenthesis, will return true:

+", "*", "!", "=", "<", ">", "@@", "#", "\$", "%", "^", "&", "|", "\\", "~", "{", "}", "[", or "]"

devu-txt-is-digit-p

(*character*: text)
 -> (*is-integer*: truth-value)

Returns true if *character* is an integer from 0 to 9, otherwise returns false.

devu-txt-is-mib-character-symbol-p

(*character*: text
-> (*is-mib-character*: truth-value)

Returns **true** if *character* is a mib character symbol, otherwise returns **false**.

The following values for *character*, enclosed in parenthesis, will return true:

{, }, (,), [,], \, |, +, =, !, *, ^, <, >, ?, |, :,
., " , "

GMIB Module

The GMIB module contains procedures for reading ASN.1 MIB's and the *trapd.conf* file.

mib-return-default-msg-category

(*mib-rec*: class mib-receiver)
-> (*default-category*: text)

Returns a value for the default-category attribute of *mib-rec* based on:

If the default-category of *mib-rec* /= "" then use this value

Else build the default-category of *mib-rec* by concatenating
[enterprise]-[genericTrap]-[specificTrap].

Caution This procedure may change the value of the default-category of *mib-rec*.

mib-substitute-in-format-spec

(*mib-rec*: class mib-receiver)
-> (*trapd-format*: text)

The Trapd-format attribute of *mib-rec* is substituted according to the rules of the format description for the *trapd.conf* file (see the *man* pages for *trapd.conf*). The return value is the fully substituted format specification.

Limitations:

- treats upper and lower case characters as the same
- "\$C" is not supported, substitutes ""****"
- "\$L" is not supported, substitutes ""****"
- "-n" and "+n" options treated as "n" option
- Only supports single digit field position on "\$n"
- Does not handle all C printf formatting

mib-receiver-create-and-queue

(*class-type*: symbol, *nodename*: text, *number-of-oids-requested*: integer)

-> (*mib-rec*: class mib-receiver)

This procedure is used in conjunction with performing non-blocking SNMP GET transactions. The user calls this procedure specifying the *class-type* of the MIB-RECEIVER object to be created. The *class-type* must be a subclass of the GET-REQUEST-RECEIVER class. This object is then placed in the MIB-RECEPTION-QUEUE to await the results of the SNMP GET transaction from the SGB process. Once the results of the SNMP GET transaction are returned from the SGB process Integrity updates the *mib-rec* object and moves it to the MIB-COMPLETED-RECEIVES queue. The *number-of-oids-requested* is an integer value specifying how many OIDs are being requested in this SNMP GET transaction. This value will be concluded to the Number-of-expected-values of the *mib-rec* object and can be compared against the Number-of-received-values that is updated by the SGB process. See the *oxs_demo.kb* non-blocking SNMP GETs for examples of its usage.

mib-create-name-to-oid-translation

(*oid-to-name-translation-obj*: class oid-to-name-translation)

Creates a NAME-TO-OID-TRANSLATION object and places it to the right of and slightly below the OID-TO-NAME-TRANSLATION object. This API can be used in a G2 procedure to create NAME-TO-OID-TRANSLATION objects for a group of OID-TO-NAME-TRANSLATION objects upon a workspace. This procedure is called by the user-menu choice create name translation for OID-TO-NAME-TRANSLATION objects.

mib-enterprise-oid-text-to-name

(*txt*: text)

-> (*resulting-name*: symbol)

Returns the Resulting-name attribute of an OID-TO-NAME-TRANSLATION object as a symbol if one exists otherwise, returns *txt* as a symbol.

mib-translate-info-to-trap-name

(*enterprise-id*: text, *generic-id*: text, *specific-id*: text)

-> (*trap-class-name*: symbol, *class-found*: truth-value)

Returns the name of the appropriate class definition, TRAP-[ENTERPRISE-ID]-[GENERIC-ID]-[SPECIFIC-ID], as a symbol with a truth value of TRUE. If no class definition can be found then the symbol not-found will be returned with a truth value of FALSE.

mib-translate-info-to-completion-name

(*enterprise-id*: text, *generic-id*: text, *specific-id*: text)

-> (*completion-routine-name*: symbol, *completion-routine-found*: truth-value)

Returns the name of the appropriate procedure, completion-[enterprise-id]-[generic-id]-[specific-id], as a symbol with a truth value of TRUE. If no

completion routine can be determined then the symbol not-found will be returned with a truth value of FALSE.

mib-delete-old-list-entries

(*lst*: class item-list, *TTL*: integer)

Deletes MIB-RECEIVER objects in the queue specified by *lst*, *mib-reception-queue* or *mib-completed-receives*, older than their Time To Live (*TTL*). The Time To Live is specified via the *mib-receiver-time-to-live-in-queues* integer-parameter. The default value for this parameter, *TTL*, is 600 seconds.

asn1-rfc1212-mib-parser

(*filename*: text, *enterprise-ws*: class item, *oid-ws*: class item, *mibrec-ws*: class item)

Reads the ASN.1 formatted MIB file specified by *filename* based on RFC 1212 definitions. The results of the read are placed on the following workspaces:

- *enterprise-ws* - Contains the ENTERPRISE-OID-TO-NAME-TRANSLATION objects
- *oid-ws* - Contains the OID-TO-NAME-TRANSLATION objects
- *mibrec-ws* - Contains the trap class definitions

mib-write-clears-file

(*enterprise-id*: text, *filename*: text, *write-mode*: symbol)

Writes the specific trap definitions “clears for” entries to the file specified by *filename*. The *write-mode* specifies whether to ‘append’ or ‘overwrite’ the file. Only MIB-RECEIVER class definitions as specified by the *enterprise-id* will be written. The value of *enterprise-id* must match the value specified in the Enterprise-id attribute of the SNMP-TRAP-RECEIVER object definition (i.e. by name such as “HP OpenView” or by dot notation such as “1.3.6.4.1.11”). A value of “*” for *enterprise-id* will include all MIB-RECEIVER class definitions.

Tip The Enterprise-id attribute value can be set by editing the Attribute-initializations of the object definition (e.g. Enterprise-id initially is “HP OpenView”).

The format of the file is as follows:

-- Header

CLEARs TRAP-[enterprise-id]-[generic-id]-[specific-id] for <specific-id>

for example:

```
-- Created at 11 Sep 97 4:34:27 p.m. by G2 for enterprise = *
CLEARs TRAP-WFSWSERIES7-XX-XX for 88888
```

or comma-separated entries such as

```
-- Created at 11 Sep 97 4:34:27 p.m. by G2 for enterprise = *
CLEARs TRAP-WFSWSERIES7-XX-XX for 88888, 99999, 77777
```

The file will be readable by the `mib-read-clears-file` procedure for use in adding Clear-attributes to MIB-RECEIVER class definitions.

mib-read-clears-file

(*filename*: text)

Reads the file specified by *filename* consisting of the 'CLEARs' entries and updates the respective trap class definition. The format of the file is as follows:

-- Header

CLEARs TRAP-[enterprise-id]-[generic-id]-[specific-id] for <specific-id>

for example:

-- Created at 11 Sep 97 4:34:27 p.m. by G2 for enterprise = *
CLEARs TRAP-WFSWSERIES7-XX-XX for 88888

or comma-separated entries such as

-- Created at 11 Sep 97 4:34:27 p.m. by G2 for enterprise = *
CLEARs TRAP-WFSWSERIES7-XX-XX for 88888, 99999, 77777

mib-trapd-preprocessed-conf-reader

(*filename*: text, *logfile*: text, *ws*: class kb-workspace, *modify-existing*: truth-value)

Reads the `trapd.conf.ppd` file specified by *filename*. This file is created from running the trapd pre-processor executable `trapd_pp` on the file `trapd.conf`. The parameter *logfile* specifies the path of the file that logs the results of the read process. The *ws* parameter specifies where the newly created trap class definitions are to be placed. The *modify-existing* parameter, true or false, specifies whether the procedure should modify existing trap class definitions.

Note Unknown enterprises will be created as **transient** OID-TO-NAME-TRANSLATIONS with a Resulting-name of "Unknown Enterprise".

mib-delete-dictionary-on-workspace

(*end-obj*: class item)

Deletes dictionary entries, OID-TO-NAME-TRANSLATION and NAME-TO-OID-TRANSLATION objects, on the workspace of *end-obj*. *end-obj*, by convention, is a message or other object placed at the bottom of the dictionary entries.

mib-clean-up-ws

(*ws*: class kb-workspace)

This procedure will attempt to delete everything, mib related, on *ws* excluding the `opfo-workspace-header`, the `opxb-go-to-superior-button`, the "Go to Bottom" action-button, and the "Delete MIB Items on this WS" action-button.

GSNMP Module

The *gsnmp.kb* module contains procedures for the simulation of SNMP traps and agent MIBs.

oxs-sim-simulate-trap

(*mib-rec*: class mib-receiver)

Simulates the reception of an SNMP trap from the SNMP Gateway Bridge. The information for the trap must be filled out in the *mib-rec* object. See [SNMP Trap Simulation](#) for additional information on using simulated SNMP traps.

oxs_sim-request-handler

(*node*: class opfo-domain-object, *snmp-type*: symbol, *node-id*: text, *machine-nodename*: text, *community-name*: text, *OID*: text; *ASN1-Type*: integer, *new-value*: text, *time-out*: integer, *request-type*: integer)
 -> (*error-code*: integer, *error-name*: symbol, *error-string*: text, *results-list*: item-or-value, *result-id*: integer)

Handles the execution of a simulated or actual SNMP transaction (SET, GET, or GET NEXT). See [SNMP Agent MIB Simulation](#) for additional information on simulated SNMP MIB transactions. This procedure makes calls to `snmp-set-rpc-call` or `snmp-get-table-column-rpc-call` for performing the remote procedure calls to the SNMP Gateway Bridge.

Argument	Description
<i>node</i>	The domain object to perform the SNMP the transaction on.
<i>snmp-type</i>	A symbolic value of <code>snmp</code> or <code>simulated-snmp</code> for specifying either a real or simulated SNMP transaction.
<i>node-id</i>	The dotted IP address of the <i>node</i> to which the transaction is directed.
<i>machine-nodename</i>	The name of the <i>node</i> to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>OID</i>	The object identifier, usually in dotted notation, for the variable involved in the SNMP transaction.

Argument	Description
<i>ASN1-Type</i>	The ASN.1 type of the variable involved in the SET transaction. Possible values are: asn1_integer (2) asn1_octet_string (4)
<i>new-value</i>	The new value of the <i>OID</i> for an SNMP SET transaction.
<i>time-out</i>	The time, in seconds, after which the non-blocking transaction will be aborted.
<i>request-type</i>	The new value of the <i>OID</i> as a text string.
Return Value	Description
<u><i>error-code</i></u>	Returns a value of 0 if the transaction is successful else the HP OpenView or NetView 6000 integer error code.
<u><i>error-name</i></u>	A symbol with values of snmp-not-loaded (the snmp modules are not loaded; possibly due to an incorrect module hierarchy), snmp-agent-not-specified (a value for <i>snmp-type</i> other than snmp or simulated-snmp was specified) or OK.
<u><i>error-string</i></u>	Returns a value of “[the current value of the time-out-interval]-[the current value of the retry-count]” if the transaction is successful else “[the text translation of the error code as defined in the HP OpenView or NetView 6000 documentation]”
<u><i>results-list</i></u>	A text list containing the results of the SNMP GET or GET NEXT transactions.
<u><i>result-id</i></u>	Returns the integer value of the <i>result-id</i> of the MIB-RECEIVER object that receives the results of the SNMP GET or GET NEXT transactions.

oxs-heartbeat-trap-procedure

(*gsi-1*: class gsi-interface, *gsi-2*: class gsi-interface)

This procedure sends a heartbeat trap to the SNMP Gateway Bridge and receives a heartbeat trap back if the SGB is up. If the SGB is down then this procedure will try to reconnect to the SGB once it is up. The *gsi-1* parameter is the synchronous SGB interface object and the *gsi-2* parameter is the event SGB interface object. This procedure is typically started from a scanning rule. See the *oxs_demo.kb* module for an example of its use.

snmp-get-mibrec-field-by-name

(*field-list*: class text-list, *nam*: text)

-> (*oid-value*: text)

This procedure will return the value whose OID translates to *nam*.

field-list should be a text list of the form: oid, value, oid, value, etc. This is normally the Results-list attribute of the MIB-RECEIVER object.

Note This is the **LEAST** efficient of the three access procedures; **snmp-get-mibrec-field-by-name**, **snmp-get-mibrec-field-by-oid**, and **snmp-get-mibrec-field-by-pos**.

snmp-get-mibrec-field-by-oid

(*field-list*: class text-list, *oid*: text)

-> (*oid-value*: text)

This procedure will return the value whose OID equals the value of the *oid* argument passed in.

field-list should be a text list of the form: oid, value, oid, value, etc. This is normally the Results-list attribute of the MIB-RECEIVER object.

snmp-get-mibrec-field-by-pos

(*field-list*: class text-list, *pos*: integer)

-> (*oid-value*: text)

This procedure will return the value in *field-list* as specified by the *pos* argument (e.g. a *pos* value of 2 will return the value of the second oid in *field-list*).

field-list should be a text list of the form: oid, value, oid, value, etc. This is normally the Results-list attribute of the MIB-RECEIVER object.

snmp-set-rpc-call

(*interface-obj*: class gsi-interface, *machine-nodename*: text, *community-name*: text, *OID*: text, *new-value*: text, *ASN1-Type*: integer)
-> (*error-code*: integer, *error-string*: text, *node-name*: text)

This procedure handles making the remote procedure call from the `oxs_sim-request-handler` procedure to the SNMP Gateway Bridge for performing an SNMP SET transaction.

Argument	Description
<i>interface-obj</i>	The asynchronous GSI object.
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>OID</i>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
<i>ASN1-Type</i>	The ASN.1 type of the variable involved in the SET transaction. Possible values are: asn1_integer (2) asn1_octet_string (4)
<i>new-value</i>	The new value of the <i>OID</i> as a text string.

Return Value	Description
<u><i>error-code</i></u>	Returns a value of 0 if the transaction is successful else the HP OpenView or NetView 6000 integer error code.
<u><i>error-string</i></u>	Returns a value of “[the current value of the time-out-interval]-[the current value of the retry-count]” if the transaction is successful else “[the text translation of the error code as defined in the HP OpenView or NetView 6000 documentation]”
<u><i>node-name</i></u>	Returns the node name to which the transaction was directed.

snmp-non-blocking-set-rpc-call

(*interface-obj*: class gsi-interface, *machine-nodename*: text,
community-name: text, *OID*: text, *req-name*: text, *new-value*: text,
ASN1-Type: integer, *Timeout*: integer)
 -> (*error-code*: integer, *error-string*: text)

This procedure return an error-code (integer, default is 99999) and error-text (text, default is "OK"). The procedure creates a mib-receiver, sets the requester-name of the mib-receiver to *req-name*, and sets the receive-completion-method of the receiver to the symbol no-method. A check is performed on the ASN1 type to call the appropriate set non-blocking RPC (integer or text). When the RPC completes, it returns a result-id. This result-id is stored in the mib-receiver. If the result-id is greater than 0, the procedure goes into a loop waiting for the return of information for the set request or until the specified *Timeout* is reached. If the *Timeout* is reached, the *error-code* is set to the value of *Timeout*. If a negative result-id is returned, the *error-code* returned is set to the result-id. This indicates the bridge could not process the set request. Before returning any values, the mib-receiver created at the beginning of this procedure, is deleted.

Argument	Description
<i>interface-obj</i>	The asynchronous GSI object.
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>OID</i>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
<i>req-name</i>	The requester's name. Can be any text.
<i>new-value</i>	The new value of the OID as a text string.
<i>ASN1-Type</i>	The ASN.1 type of the variable involved in the SET transaction. Possible values are: asn1_integer (2) asn1_octet_string (4)
<i>Timeout</i>	The number of seconds to wait for a response from the SET operation for this procedure. This value has no effect on the timeout value for the SNMP SET request.

Return Value	Description
<i>error-code</i>	Returns a value less than 0 if the G2-SNMP bridge could not process the SET request, a value of 99999 indicates successful operation, and a value equal to that of the specified <i>Timeout</i> indicates no response from the SNMP agent.
<i>error-string</i>	Returns a value of "[the name of this procedure] received a [result-id] return value from SET_NONBLOCKING_SINGLE" if the transaction is successful else "[the name of this procedure] timed out waiting for a response for the mib-receiver"

snmp-get-rpc-call

(*interface-obj*: class *gsi-interface*, *machine-nodename*: text, *community-name*: text, *OID*: text, *transaction-tagname*: text, *time-out*: integer)
 -> (*error-code*: integer, *error-string*: text, *node-name*: text, *results-list*: class *item-list*)

This procedure handles making the remote procedure call, from the *oxs_sim-request-handler* procedure, to the SNMP Gateway Bridge for performing an SNMP GET transaction.

Argument	Description
<i>interface-obj</i>	The asynchronous GSI object.
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.
<i>OID</i>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
<i>transaction-tagname</i>	A handle for matching an incoming response with the outgoing request.
<i>time-out</i>	The time, in seconds, before Integrity gives up on the SNMP Gateway Bridge returning the results of the GET transaction.

Return Value	Description
<i><u>error-code</u></i>	Returns a value of 0 if the transaction is successful else the HP OpenView or NetView 6000 integer error code.
<i><u>error-string</u></i>	Returns a value of “[the current value of the time-out-interval]-[the current value of the retry-count]” if the transaction is successful else “[the text translation of the error code as defined in the HP OpenView or NetView 6000 documentation]”
<i><u>node-name</u></i>	Returns the node name to which the transaction was directed.
<i><u>results-list</u></i>	The Results-list of the MIB-RECEIVER object that receives the results of the SNMP GET transaction. The values are stored in this list as OID/Value pairs. Use the procedures <code>snmp-get-mibrec-field-by-name</code> , <code>snmp-get-mibrec-field-by-oid</code> , and <code>snmp-get-mibrec-field-by-pos</code> to retrieve values from this list.

snmp-get-table-column-rpc-call

(*interface-obj*: class `gsi-interface`, *machine-nodename*: text, *community-name*: text, *OID*: text, *transaction-tagname*: text, *time-out*: integer)
 -> (*error-code*: integer, *error-string*: text, *node-name*: text, *results-list*: item-or-value)

This procedure handles making the remote procedure call from the `oxs_sim-request-handler` procedure to the SNMP Gateway Bridge for performing an SNMP GET table column transaction. This procedure will traverse the table and return the results in the *results-list*.

Argument	Description
<i>interface-obj</i>	The asynchronous GSI object.
<i>machine-nodename</i>	The name or dotted IP address of the node to which the transaction is directed.
<i>community-name</i>	The administrative relationship for the transaction.

Argument	Description
<i>OID</i>	The object identifier, usually in dotted notation, for the variable involved in the GET transaction.
<i>transaction-tagname</i>	A handle for matching an incoming response with the outgoing request.
<i>time-out</i>	The time, in seconds, before Integrity gives up on the SNMP Gateway Bridge returning the results of the GET Table Column transaction.

Return Value	Description
<u><i>error-code</i></u>	Returns a value of 0 if the transaction is successful else the HP OpenView or NetView 6000 integer error code.
<u><i>error-string</i></u>	Returns a value of “[the current value of the time-out-interval]-[the current value of the retry-count]” if the transaction is successful else “[the text translation of the error code as defined in the HP OpenView or NetView 6000 documentation]”
<u><i>node-name</i></u>	Returns the node name to which the transaction was directed.
<u><i>results-list</i></u>	The Results-list of the MIB-RECEIVER object that receives the results of the SNMP GET transaction. The values are stored in this list as OID/Value pairs. Use the procedures <code>snmp-get-mibrec-field-by-name</code> , <code>snmp-get-mibrec-field-by-oid</code> , and <code>snmp-get-mibrec-field-by-pos</code> to retrieve values from this list.

Functions

devu-ext-name-or-name-as-string

(*domain-object*: class [opfo-managed-object |opfo-containment-object])

-> (*external-name*: text)

Used to retrieve a text identification for an object. This function tries to find the text identification in the following order until an identification is found:

- The *opfo-external-name* of *domain-object*.
- The text returned by the function defined in the G2 parameter *devu-user-name-as-string* if one is defined. This allows you to define an alternate method to identify domain objects. You can define a custom *devu-user-name-as-string* function using initialization.
- The G2 name that is assigned to *domain-object* as a text string, if there is a name assigned. It is not recommended that you use G2 names for identifying objects since special characters are not easily supported in the G2 name of an object.
- Text describing *domain-object*. For example, “the item of <class> located on the workspace <Workspace>”

If you define a custom function in the G2 parameter *devu-user-name-as-string*, you must also be sure to define a new procedure to lookup an object when the identification text is passed to *devu-domain-object-lookup*. The G2 parameter *devu-alternate-object-lookup* provides the name of the alternate procedure.

is-a-message

(*test-item*: class item)

-> (*is-message*: truth-value)

Returns true if *test-item* is a message or message subclass, else returns false. This function can be passed in as the *ok-to-traverse* argument to the procedures *net-traversal* and *net-traversal-and-collect* to make sure that only messages are added to the list of traversed items.

is-an-object

(*test-item*: class item)

-> (*is-object*: truth-value)

Returns true if *test-item* is an object or an object subclass, else returns false. This function can be passed in as the *ok-to-traverse* argument to the procedures *net-traversal* and *net-traversal-and-collect* to make sure that only objects are added to the list of traversed items.

mib-oid-strip-instance-number

(*OID*: text)

-> (*resulting-name*: symbol)

Returns a symbol of the **Resulting-name** attribute of an **OID-TO-NAME-TRANSLATION** object. If no **OID-TO-NAME-TRANSLATION** object exists this function returns the symbol **no-name**. The **OID-TO-NAME-TRANSLATION** object is found by stripping off the last number of the *OID* string and then checking for the existence of an **OID-TO-NAME-TRANSLATION** object. If no **OID-TO-NAME-TRANSLATION** object exists the process is repeated until the *OID* string is empty.

Caution This is a recursive function and for an extremely long *OID* the recursion limit of G2 could be exceeded. The default limit for G2 is 50, see the *G2 Reference Manual* for information on resetting the recursion limit value.

mib-oid-symbol-to-name

(*OID*: symbol)

-> (*resulting-name*: symbol)

Searches for an **OID-TO-NAME-TRANSLATION** object named by *OID* (i.e. an **OID** string such as 1.3.6.4.1.11.2.3.1.1.1) where *OID* is a symbolic value. If one is found then it returns the **Resulting-name** attribute of the object as a symbol else it returns the symbol **no-name**.

mib-oid-text-to-name

(*OID*: text)

-> (*resulting-name*: symbol)

Searches for an **OID-TO-NAME-TRANSLATION** object named by *OID* (i.e. an **OID** string such as 1.3.6.4.1.11.2.3.1.1.1) where *OID* is a text value. If one is found then it returns the **Resulting-name** attribute of the object as a symbol else it returns the symbol **no-name**.

mib-default-trapd-priority-conversion

(*input-value*: integer)

-> (*priority*: integer)

Returns an integer priority value (as defined in Integrity) converted from the priority *input-value* (as defined in HP OpenView), as follows:

input-value	priority
1	6
2	4
3	3
4	2

input-value	priority
5	1
any value	1

snmp-severity-to-status-text-conversion*(input-value: integer)**-> (priority-string: text)*Returns a text string of the priority *input-value*, as follows:

input-value	priority-string
1	Critical
2	Major
3	Minor
4	Warning
6	Normal
99999	Unknown
any value	Normal

snmp-desc-value*(mib-rec: class mib-receiver)**-> (trap-description: text)*

Returns a text string of the Generic-trap attribute of a MIB-RECEIVER object, as follows:

the generic-trap of mib-rec	trap-description
0	Cold Start
1	Warm Start
2	Link Down
3	Link Up
4	Authentication Failure
5	EGP Neighbor Loss
any value	Received [the default-category of <i>mib-rec</i>]

Reporting Errors

Describes how to report bugs in a G2-SNMP Bridge to Gensym customer support.



If you encounter a bug in the G2-SNMP Bridge, please collect as much information as possible concerning the circumstances of the failure before calling Gensym Customer Support. Such information should include, but is not limited to the:

- SNMP Gateway Bridge version you are using. This information is displayed in the terminal window from which you first start the SGB. If you do not wish to start the SGB, you may execute it with the `-v` option, which prints out only the SGB version information. For example:

At the UNIX prompt, type:

```
% g2snmpov.41 -v
```

The SGB prints out:

```
Gensym G2-SNMP HP OpenView Bridge for Solaris
```

```
Version 2.3 Rev 0 Build f01
```

```
This software is compatible with Integrity Version 5.0 Rev. 0
```

The first two lines of this printout provide the SGB version information needed by Gensym Customer Support.

- Platform operating system version.
- Version of HP OpenView if the SNMP Gateway Bridge is being used with HP OpenView.
- Integrity version.
- G2 version.

- Type of operation being performed: trap sent/received, SNMP get or set request.
- Number of G2 processes connected to the SNMP Gateway Bridge.
- Any error messages printed out by the SNMP Gateway Bridge or other processes, including messages displayed on the G2 message board or in the Operator Logbook.
- Any unusual conditions such as a trap storm or abnormally high network traffic.

Notify Gensym Customer Support. See [Customer Support Services](#).

APIs and Initializations

Chapter 25: Core Services APIs

Provides information on Integrity Core Services APIs.

Chapter 26: OPAC APIs

Provides information on the Integrity OPAC APIs.

Chapter 27: Startup Parameters

Provides information on Integrity Initializations.

Core Services APIs

Provides a listing of all of the Integrity Core Services APIs: procedures, methods, and functions.

Introduction **511**

Procedures Listed by Module **512**

GND0 Module **513**

GLF Module **529**

Functions **530**

Methods Listed by Class **532**



Introduction

This chapter describes the OPAC Core Services APIs.

Procedures Listed by Module

The API procedures are listed according to the module that contains them. The modules that include API procedures include:

- [GNDO Module](#) - Core Network Domain module that contains procedures for parsing and looking up domain objects, networking, message handling, browser and status bar, and Telewindows.
- [GLF Module](#) - G2 Log File module that contains all the procedures used to implement the logging functions.

GNDO Module

The GNDO module contains procedures for parsing and looking up domain objects, networking, message handling, browser and status bar, and Telewindows.

devu-consume-next-field

(*txt*: text, *separator*: text)
-> *next-field*, *text-remaining*: text

Procedure to parse a text string containing fields separated by a unique separator text. Returns two text strings. *next-field* contains *txt* up to, but not including, the first occurrence of *separator*. *text-remaining* contains what is left in *txt* after *next-field* is stripped off.

devu-decode-comma-line

(*text-string*: text, *txt-list*: class text-list)

Use this procedure to turn a comma separated text string into a list of text items. Breaks comma separated strings in *text-string* into individual strings and inserts them into the list *txt-list* without any duplicate items.

devu-domain-object-lookup

(*object-name*: text)
-> *domain-object*: class [opfo-managed-object | opfo-containment-object]

This procedure looks for the domain object several different ways:

- 1 If a procedure is specified for the parameter `devu-alternate-object-lookup-procedure`, then `devu-domain-object-lookup` calls this custom procedure and passes it *object-name*. The custom procedure must return a domain object. You define the name of your custom procedure in `devu-alternate-object-lookup-procedure`, using an initialization item.
- 2 If no custom lookup procedure is defined, `devu-domain-object-lookup` looks for a domain object with the `opfo-external-name` *object-name*. If none is found, the procedure looks for an object with a G2 name *object-name*.
- 3 Finally, if no domain object can be found to match *object-name*, the procedure calls the procedure named by `devu-unknown-object-procedure`. The default procedure defined is `devu-default-object-return` which returns the object `devu-safe-object`. If you want to change the default object returned, you can modify the initialization for `devu-unknown-object-procedure`.

devu-error-handler

(*target*: class item, *sender*: class item, *error-category*: text,
priority: integer, *error-name*: symbol, *error-text*: text, *lifetime*: integer)

This is the default error handler in Integrity. This procedure executes the error handler procedure specified in the G2 parameter `devu-error-handler-proc`.

You can define a custom error handler procedure and specify the name of the procedure, using the initialization for `devu-error-handler-proc`.

`devu-get-field-n`

(*txt*: text, *n*: integer, *separator*: text)

-> *nth-field*: text

The argument *txt* is a text string containing fields separated by a *separator* text. This procedure returns the *n*th field in *txt*. If *txt* does not have *n* fields, an empty string is returned.

`devu-insert-item-in-sorted-ascending-list`

(*new-item*: class item, *list*: class item-list, *key-proc*: class procedure)

Inserts *new-item* in *list*, in ascending order based on criteria defined in *key-proc*. The procedure defined by *key-proc* should accept an `item` as an argument and return an integer used for sorting the items.

`devu-insert-item-in-sorted-descending-list`

(*new-item*: class item, *list*: class item-list, *key-proc*: class procedure)

Inserts *new-item* in *list*, in descending order based on criteria defined in *key-proc*. The procedure defined by *key-proc* should accept an `item` as an argument and return an integer used for sorting the items.

`devu-insert-msg-in-alphabetical-list`

(*msg*: class message, *list*: class item-list)

Inserts a *msg* in alphabetical order into *list*. No duplicates are allowed in the list.

`devu-insert-msg-in-alphabetical-list-allowing-duplicates`

(*msg*: class message, *list*: class item-list)

Inserts a *msg* in alphabetical order into *list*. Duplicates are allowed in the list.

`devu-pattern-matcher-with-indices`

(*txt*: text, *txt-pos*: integer, *pattern*: text, *pattern-pos*: integer)

-> *match-value*: truth-value

Searches for *pattern* in *txt*. The symbol "*" in *pattern* is matched by any number of arbitrary characters. The symbol "?" in *pattern* is matched by exactly one arbitrary character. The characters "*?" together are not matched. The match begins starting from the position *pattern-pos* in the pattern and *txt-pos* in the text string *txt*. The procedure returns `true` if a match is found, otherwise `false`.

`devu-safe-text-for-symbol`

(*txt*: text)

-> *txt-to-symbol*: text

Converts a text string to a valid G2 symbol string. It replaces all spaces with "_", and prefixes all special characters with "@@". Spaces at the end of the string are removed.

devu-safe-window

()
-> *safe-window*: class g2-window

Returns a safe G2 window.

devu-strip-linefeeds

(*txt*: text)
-> *txt-without-linefeeds*: text

Strips the linefeeds from *txt* and returns the text without any linefeeds.

devu-substitute-text

(*match-string*: text, *replace-string*: text, *txt*: text)
-> *new-txt*: text

Finds every occurrence of *match-string* in *txt* and replaces it with *replace-string*. The new text string is returned in *new-txt*.

devu-text-is-ascii-punctuation-p

(*txt*: text)
-> *is-symbol*: truth-value

Returns true if *txt* is an ascii punctuation; else returns false.

devu-text-is-ascii-symbol-p

(*txt*: text)
-> *is-symbol*: truth-value

Returns true if *txt* is an ascii symbol; else returns false.

devu-txt-is-digit-p

(*character*: text)
-> *is-integer*: truth-value

Returns true if *character* text is an integer from 0 to 9; otherwise returns false.

devu-txt-is-mib-character-symbol-p

(*character*: text)
-> *is-mib-character*: truth-value

Returns true if *txt* is a mib-character-symbol; otherwise returns false.

do-nothing-at-node-collection-visit

(*node*: class item, *mgr*: class item, *bag*: class item)

This procedure is an example of a procedure that can be passed as the *node-visit-proc* argument of the procedure [net-traversal-and-collect](#) (*node*: class item, *child-finder*: class procedure, *ok-to-traverse*: class function-definition, *node-visit-proc*: class procedure, *search-type*: symbol, *mgr*: class item, *list*:

[class devu-item-list-without-duplicates, bag: class item](#)) or [net-traversal-and-collect-2 \(node: class item, child-finder-2: class procedure, ok-to-traverse-2: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, list: class devu-item-list-without-duplicates, bag: class item\)](#). Whenever an item is found and placed on the net traverse list, the procedure defined in the *node-visit-proc* argument is called.

This procedure differs from *do-nothing-at-node-visit* only in that it has an extra argument, *bag*, to gather additional information at a visit. *bag* is usually an item list without duplicate elements. This is required by the procedures that add collection to their traversal process.

This procedure is provided as a template for a custom procedure to be called by those routines.

For more information see [net-traversal \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates\)](#).

do-nothing-at-node-visit

(*node: class item, mgr: class item*)

This procedure is an example of a procedure that can be passed as the *node-visit-proc* argument of the procedure [net-traversal \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates\)](#) or [net-traversal-2 \(node: class item, child-finder-2: class procedure, ok-to-traverse-2: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, list: class devu-item-list-without-duplicates\)](#). Whenever an item is found and placed on the net traverse list, the procedure defined in the *node-visit-proc* argument is called.

The procedure is provided as a template for a custom procedure to be called by those routines.

For more information see [net-traversal \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates\)](#).

find-connected-children

(*parent: class object, ok-to-traverse: class function-definition, children: class devu-item-list-without-duplicates, list: class devu-item-list-without-duplicates*)

This procedure is an example of a procedure that can be passed as the *child-finder* argument of the procedure [net-traversal \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list:](#)

[class devu-item-list-without-duplicates](#)) or [net-traversal-2](#) ([node: class item](#), [child-finder-2: class procedure](#), [ok-to-traverse-2: class function-definition](#), [node-visit-proc: class procedure](#), [search-type: symbol](#), [mgr: class item](#), [list: class devu-item-list-without-duplicates](#)). This procedure defines a criteria for selecting objects during a net traversal.

`ok-to-traverse` is the name of a function that takes in an object as argument, checks a criteria for net traversal, and returns true or false. The *list* argument is used as a filter. The [find-connected-children](#) ([parent: class object](#), [ok-to-traverse: class function-definition](#), [children: class devu-item-list-without-duplicates](#), [list: class devu-item-list-without-duplicates](#)) procedure finds all the objects connected to the parent object that satisfy the criteria specified by the `ok-to-traverse` function and that are not already a member of *list*, and inserts them at the end of the children list.

For more information, see [net-traversal](#) ([node: class item](#), [child-finder: class procedure](#), [ok-to-traverse: class function-definition](#), [node-visit-proc: class procedure](#), [search-type: symbol](#), [mgr: class item](#), [visited-node-list: class devu-item-list-without-duplicates](#)).

find-downstream-connected-children

(*parent: class object*, *ok-to-traverse: class function-definition*,
children: class devu-item-list-without-duplicates,
list: class devu-item-list-without-duplicates)

This procedure is an example of a procedure that can be passed as the *child-finder* argument of the procedure `net-traversal` or `net-traversal-and-collect`. This procedure defines a criteria for selecting an object during a net traversal.

`ok-to-traverse` is the name of a function that takes in an object as argument, checks a criteria for net traversal and returns true or false. The *list* argument is used as a filter. `find-downstream-connected-children` finds all the objects connected to the output port of the parent object that satisfy the criteria specified by the `ok-to-traverse` function and that are not already a member of *list*, and inserts them at the end of the children list.

For more information, see [net-traversal](#) ([node: class item](#), [child-finder: class procedure](#), [ok-to-traverse: class function-definition](#), [node-visit-proc: class procedure](#), [search-type: symbol](#), [mgr: class item](#), [visited-node-list: class devu-item-list-without-duplicates](#)).

find-downstream-connected-children-2

(*parent: class object*, *mgr: class item*,
ok-to-traverse-2: class function-definition,
children: class devu-item-list-without-duplicates,
list: class devu-item-list-without-duplicates)

This procedure is an example of a procedure that can be passed as the *child-finder* argument of the procedure [net-traversal-2](#) ([node: class item](#), [child-finder-2: class procedure](#), [ok-to-traverse-2: class function-definition](#), [node-](#)

[visit-proc: class procedure, search-type: symbol, mgr: class item, list: class devu-item-list-without-duplicates](#)) or [net-traversal-and-collect-2 \(node: class item, child-finder-2: class procedure, ok-to-traverse-2: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, list: class devu-item-list-without-duplicates, bag: class item\)](#). This procedure defines a criteria for selecting an object during a net traversal.

It differs from [find-downstream-connected-children \(parent: class object, ok-to-traverse: class function-definition, children: class devu-item-list-without-duplicates, list: class devu-item-list-without-duplicates\)](#) in that it is also passed *mgr* by the net traversal procedure.

ok-to-traverse-2 is the name of a function that takes in an object and an item, *mgr*, as its arguments, checks a criteria for the object for net traversal and returns true or false. The *list* argument is used as a filter. The [find-downstream-connected-children-2 \(parent: class object, mgr: class item, ok-to-traverse-2: class function-definition, children: class devu-item-list-without-duplicates, list: class devu-item-list-without-duplicates\)](#) procedure finds all the objects that are connected to an output port of the parent that satisfy the criteria specified by the *ok-to-traverse-2* function and that are not already a member of *list*, and inserts them at the end of the children list. New items are added to the end of the list.

For more information, see [net-traversal \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates\)](#).

find-subworkspace-children

(*parent: class object, ok-to-traverse: class function-definition, children: class devu-item-list-without-duplicates, list: class devu-item-list-without-duplicates*)

This procedure is an example of a procedure that can be passed as the *child-finder* argument of the procedure [net-traversal \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates\)](#) or [net-traversal-and-collect \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, list: class devu-item-list-without-duplicates, bag: class item\)](#). This procedure defines a criteria for selecting an object during a net traversal.

ok-to-traverse is the name of a function that takes in an object as argument, checks a criteria for net traversal and returns true or false. The *list* argument is used as a filter. The [find-subworkspace-children \(parent: class object, ok-to-traverse: class function-definition, children: class devu-item-list-without-duplicates, list: class devu-item-list-without-duplicates\)](#) procedure finds all the objects located on the subworkspace of parent that satisfy the criteria specified

by the `ok-to-traverse` function and that are not already a member of *list*, and inserts them at the end of the children list.

For more information, see [net-traversal \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates\)](#).

find-upstream-connected-children

(*parent*: class object, *ok-to-traverse*: class function-definition,
children: class devu-item-list-without-duplicates,
list: class devu-item-list-without-duplicates)

This procedure is an example of a procedure that can be passed as the *child-finder* argument of the procedure [net-traversal \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates\)](#) or [net-traversal-and-collect \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, list: class devu-item-list-without-duplicates, bag: class item\)](#). This procedure defines a criteria for selecting an object during a net traversal.

`ok-to-traverse` is the name of a function that takes in an object as argument, checks a criteria for net traversal and returns true or false. The *list* argument is used as a filter. The [find-upstream-connected-children \(parent: class object, ok-to-traverse: class function-definition, children: class devu-item-list-without-duplicates, list: class devu-item-list-without-duplicates\)](#) procedure finds all the objects connected to the output port of the parent that satisfy the criteria specified by the `ok-to-traverse` function and that are not already a member of *list*, and inserts them at the end of the children list.

For more information, see [net-traversal \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates\)](#).

net-traversal

(*node*: class item, *child-finder*: class procedure,
ok-to-traverse: class function-definition, *node-visit-proc*: class procedure,
search-type: symbol, *mgr*: class item,
visited-node-list: class devu-item-list-without-duplicates)

Use to traverse a network starting from *node* and find a list of visited items related to node that satisfy a specified criteria. The argument *child-finder* procedure defines the type of relationship between the items. Examples of relationships are: connected items, items connected at the input of node, and items contained in node. The argument *ok-to-traverse* function defines the criteria for items to find. *node-visit-proc* is a procedure called whenever a

node is visited. *search-type* defines the type of search, depth-first or breadth-first. The default is breadth-first.

The related items found to satisfy the criteria are placed on the list *visited-node-list*. If this list contains items when it is passed as an argument, the items found will be added to the end of the list and no duplicate items will be added to the list. The items passed in on *visited-node-list* filter out those items and the items related to them in the search.

You can define your own procedures and functions for *child-finder*, *ok-to-traverse* and *node-visit-proc*, or you can use procedures and functions defined in the system. The defined system procedures and functions that can be passed as arguments to *net-traversal* are listed in the table below:

Procedure or Function Name	Description
<i>find-connected-children</i>	Finds all objects connected to <i>node</i> .
<i>find-downstream-connected-children</i>	Finds all objects connected at an output of the <i>node</i> item.
<i>find-subworkspace-children</i>	Finds all items contained on the subworkspace of the <i>node</i> item.
<i>find-upstream-connected-children</i>	Finds all objects connected at an input of the <i>node</i> item.
<i>do-nothing-at-node-visit</i>	Sample procedure does nothing at node visit.
<i>is-an-object</i>	Passed as <i>ok-to-traverse</i> function. Tests to see if item is an object.
<i>is-a-message</i>	Passed as <i>ok-to-traverse</i> function. Tests to see if item is a message.

These procedures are documented in this chapter. They are examples of procedures that can be used to customize the way [net-traversal \(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates\)](#) works. These are public procedures that can be used as templates for your own special purpose procedures.

mgr is not used directly by *net-traversal*. It is there for the user, for any needed purpose, such as containing additional context or for accumulating results of a search. *mgr* is passed to the *node-visit-proc*.

Note that the starting node is added to the *visited-node-list*. The list implies the order of the nodes visited. The traversal can start with a node that does not satisfy the *ok-to-traverse* criteria.

Three other net traversal routines are provided. The differences between these and *net-traversal* are outlined below:

[net-traversal-and-collect](#) (*node*: class item, *child-finder*: class procedure, *ok-to-traverse*: class function-definition, *node-visit-proc*: class procedure, *search-type*: symbol, *mgr*: class item, *list*: class devu-item-list-without-duplicates, *bag*: class item) - The procedure defined by *node-visit-proc* is passed an additional parameter to let you collect additional information when you visit an item.

[net-traversal-2](#) (*node*: class item, *child-finder-2*: class procedure, *ok-to-traverse-2*: class function-definition, *node-visit-proc*: class procedure, *search-type*: symbol, *mgr*: class item, *list*: class devu-item-list-without-duplicates) - *mgr* is passed to the *child-finder* procedure and the *ok-to-traverse* function.

[net-traversal-and-collect-2](#) (*node*: class item, *child-finder-2*: class procedure, *ok-to-traverse-2*: class function-definition, *node-visit-proc*: class procedure, *search-type*: symbol, *mgr*: class item, *list*: class devu-item-list-without-duplicates, *bag*: class item) - *mgr* is passed to the *child-finder* procedure and the *ok-to-traverse* function. Also, the procedure defined by *node-visit-proc* is passed an additional parameter to let you collect additional information when you visit an item.

To view the sample network traversal procedures:

- 1 Using the Finder, locate the netu-top-level Workspace.
- 2 Right-click on workspace and select View Item. The workspace appears on the desktop.
- 3 Choose Netu Public.

net-traversal-2

(*node*: class item, *child-finder-2*: class procedure, *ok-to-traverse-2*: class function-definition, *node-visit-proc*: class procedure, *search-type*: symbol, *mgr*: class item, *list*: class devu-item-list-without-duplicates)

Identical to *net-traversal*, except that *mgr* is passed to the *child-finder-2* routine and to the *ok-to-traverse-2* function, so that more context is available when generating these children. Note that you must use a different set of *child-finder* procedures and *ok-to-traverse* functions since an extra argument is required.

The system procedures defined which can be passed as arguments to *net-traversal-2* are listed in the table below:

Procedure name	Description
find-downstream-connected-children-2 (parent: class object, mgr: class item, ok-to-traverse-2: class function-definition, children: class devu-item-list-without-duplicates, list: class devu-item-list-without-duplicates)	Finds all objects connected at an output of the <i>node</i> item.
do-nothing-at-node-visit (node: class item, mgr: class item)	Called by procedure but does not do anything at node visit.

These procedures are provided as templates. See [net-traversal](#) (node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates) for more details.

net-traversal-and-collect

(node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, list: class devu-item-list-without-duplicates, bag: class item)

Identical to [net-traversal](#) (node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates), except that *node-visit-proc* is additionally passed a list *bag*. An example of how you might use this would be to collect a list of all messages associated with the nodes traversed in *bag*.

See [net-traversal](#) (node: class item, child-finder: class procedure, ok-to-traverse: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, visited-node-list: class devu-item-list-without-duplicates) for details on the procedures and functions that can be passed in as arguments to *net-traversal-and-collect*.

net-traversal-and-collect-2

(node: class item, child-finder-2: class procedure, ok-to-traverse-2: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, list: class devu-item-list-without-duplicates, bag: class item)

This procedure is identical to [net-traversal-2](#) (node: class item, child-finder-2: class procedure, ok-to-traverse-2: class function-definition, node-visit-proc: class procedure, search-type: symbol, mgr: class item, list: class devu-item-

[list-without-duplicates](#)) except that *node-visit-proc* is additionally passed a list, *bag*. An example of how this may be used would be to collect a list of all messages associated with the nodes traversed in *bag*.

The system functions and procedures defined which can be passed as arguments to *net-traversal-2* are listed in the table below:

Procedure or Function Name	Description
find-downstream-connected-children-2 (parent: class object , mgr: class item , ok-to-traverse-2: class function-definition , children: class devu-item-list-without-duplicates , list: class devu-item-list-without-duplicates)	Finds all objects connected at an output of the <i>node</i> item.
do-nothing-at-node-collection-visit (node: class item , mgr: class item , bag: class item)	Called by procedure but does not do anything at node visit.

These procedures and functions are provided as templates. See [net-traversal](#) ([node: class item](#), [child-finder: class procedure](#), [ok-to-traverse: class function-definition](#), [node-visit-proc: class procedure](#), [search-type: symbol](#), [mgr: class item](#), [visited-node-list: class devu-item-list-without-duplicates](#)) for more details.

sc-toggle

(*button*: class sc-toggle-button)
-> *toggle-value*: truth-value

This API toggles *button* and returns true if the button is toggled-on; otherwise false.

smh-acknowledgement-proc

(*msg*: class smh-transient-message, *win*: class object)

Sets the acknowledgment status of *msg* to "acknowledged".

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-object*: class smh-small-message

Creates a new message. The table below describes the arguments:

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 UNIX-time format. To force this to be set to the current time, pass in any negative number.
<i>priority</i>	Priority of the message.
<i>lifetime</i>	Time in seconds the message is maintained before being deleted.
<i>show-display</i>	Not currently used.

Argument	Description
<i>win</i>	Any object or G2 window. Currently not used.
<i>options</i>	<p>-nohist – Do not maintain a history for the message.</p> <p>-nack – Set the message as an acknowledged message.</p> <p>Several options define how to handle a new message that is a duplicate of an existing message. Only one of these may 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 repetition counter 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.</p>

smh-delete-all-history

()

Deletes all the message histories.

smh-delete-history

(*target*: class object, *sender*: item-or-value, *category*: text, *message-category-starting-position*: integer)

Deletes the message histories that matches the *target*, *sender*, and *category*. To match any sender, define *sender* to be the symbol **any-sender**. The characters "*" and "?" can be used to match patterns in the *category* argument. The symbol "*" matches any number of unspecified characters. The symbol "?" matches exactly one unspecified character. The *message-category-starting-position* defines a starting position from which to match the *category* to the message category of the history.

smh-delete-message

(*server*: class smh-message-server, *sender*: class object, *target*: class object, *category*: text, *win*: class object, *options*: text)

Use this procedure to delete a message from the specified *server* or, if the option “-all” is passed, from all message servers. The message is specified by defining the *sender*, *target*, and *category* of the message. *win* can be any G2 window or object.

If you can pass the message object directly, you can delete a message using [smh-message-delete-proc](#) (*msg*: class smh-transient-message, *win*: class object). To delete all messages in a message server, use [smh-server-delete-all-msg](#) (*server*: class smh-message-server).

smh-delete-server

(*server*: class: smh-message-server)

Use this procedure to delete *server* and all of its messages.

smh-get-message-history

(*target*: class object, *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)

Use this procedure to perform a historical query. The arguments for this procedure are shown in the table below:

Argument	Description
<i>target</i>	Target of the message.
<i>sender</i>	Sender of the message. The symbol any-sender 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 “?”.
<i>message-category-starting-position</i>	Starting position in the category of the history that you want to match with the <i>category</i> argument.
<i>timestamp-now</i>	Time from which you want to search backwards for messages. Passing in a negative integer selects the current time.

Argument	Description
<i>match-time-interval-seconds</i>	How far back you want to search the history. This is expressed in seconds and is measured going back from <i>timestamp-now</i> .
<i>time-stamps-list</i>	Empty list provided to hold the results of the history query as timestamps.

For more information about searching message histories, see the “Querying Message Histories” section in the *Integrity User’s Guide*.

smh-get-messages-about

(*target*: opfo-domain-object, *msg-list*: class item-list)

Use this procedure to place all the messages that target the domain object *target* into the item list *msg-list*.

smh-get-messages-in-server

(*server*: smh-message-server, *msg-list*: class item-list)

Use this procedure to place all the messages in *server* into the item list *msg-list*.

smh-get-messages-sent-by

(*sender*: object, *msg-list*: class item-list)

Use this procedure to place all the messages sent by *sender* into the item list *msg-list*.

smh-message-delete-proc

(*msg*: class smh-transient-message, *win*: class object)

Use this procedure to delete a message. *win* can be any object or G2 window. To delete a message with a particular *target*, *sender*, and *category*, use the procedure [smh-delete-message](#) (*server*: class smh-message-server, *sender*: class object, *target*: class object, *category*: text, *win*: class object, *options*: text). To delete all messages in a particular message server, use the procedure [smh-server-delete-all-msg](#) (*server*: class smh-message-server).

smh-message-query

(*target*: opfo-domain-object, *sender*: item-or-value, *category*: text, *category-match-starting-position*: integer, *msg-list*: class item-list)

Use this procedure to place all the messages that match the specified *target*, *sender*, and *category* into the item list *msg-list*. You specify *category* by using the wildcards “*”, which match any number of characters and “?”, which matches exactly one character. The argument *category-match-starting-position* specifies the position in the message category at which to start the match for *category*.

smh-propagate-message-text

(*msg*: class smh-transient-message, *new-text*: text)

Updates the message text to *new-text*. Note that the text of the message displayed on the browser will get updated only if you use this procedure.

smh-read-server-mib-from-file

(*server*: class smh-message-server, *filename*: text)

Use this procedure to read message data from a text file and create and send those messages into the specified *server*. You can create the message data text file, using the procedure [smh-write-server-mib-to-file \(server: class smh-message-server, filename: text\)](#).

smh-send-error-message

(*target*: class item, *sender*: class item, *error-category*: text, *priority*: integer, *error-name*: symbol, *error-text*: text, *error-lifetime*: integer)

This is the default `devu-error-handler-proc`. When you call the procedure `devu-error-handler`, this procedure is called by default. This procedure creates an error message and sends it to the error message server defined by the G2 parameter `smh-system-error-server`. The default error message server is `smh-error-server`.

For a detailed explanation of error handling and how it can be customized, see the “Error Handling” section of the *Integrity User’s Guide*.

smh-server-delete-all-msg

(*server*: class smh-message-server)

Use this procedure to delete all messages in *server*. To delete a message by passing the message object as an argument, use the procedure [smh-message-delete-proc \(msg: class smh-transient-message, win: class object\)](#). To delete a message that matches a specified *target*, *sender* and *category*, use the procedure `smh-delete-message`.

smh-write-server-mib-to-file

(*server*: class smh-message-server, *filename*: text)

Writes the data of the messages in the *server* to the file *filename*. The message data can be read back and turned into messages, using the procedure [smh-read-server-mib-from-file \(server: class smh-message-server, filename: text\)](#).

twm-hide-screen

(*wksp*: class kb-workspace, *win*: class g2-window)

Use this procedure to hide a workspace displayed by using the method `twm-display-screen` on a window. To get the handle of the window, *win*, displaying *wksp*, use the following statement:

```
win = the window that is twm-showing-wksp wksp;
```

GLF Module

The GLF module is the G2 Log File module. It contains all the procedures used to implement the logging functions.

glf-disable-logging

(*log*: class glf-logging-manager, *client*: class object)

Disables the logging and closes the current log file of *log*.

glf-enable-logging

(*log*: class glf-logging-manager, *client*: class object)

Enables logging and opens a new log file for *log*.

glf-set-fixed-log-closing-times

(*log*: class glf-logging-manager, *closing-time-list*: class integer-list, *client*: class object)

Use this procedure to set the daily closing schedule of the log files. *closing-time-list* is an integer list. A list of closing times is specified to allow you to close the existing log file and reopen a new file more than one time a day. The time is specified as the number of minutes since 12 A.M.

glf-write-to-log-file

(*log*: class glf-logging-manager, *log-text*: text, *client*: class object)

Writes *log-text* to the current log file of *log*.

Functions

The Integrity API includes the following functions:

[devu-ext-name-or-name-as-string](#) (*domain-object*: class [[opfo-managed-object](#) | [opfo-containment-object](#)]) -> *external-name*: text

[is-a-message](#) (*test-item*: class item) -> *is-message*: truth-value

[is-an-object](#) (*test-item*: class item) -> *is-object*: truth-value

devu-ext-name-or-name-as-string

(*domain-object*: class [[opfo-managed-object](#) | [opfo-containment-object](#)])
-> *external-name*: text

Used to retrieve a text identification for an object. This function tries to find the text identification in the following order until one is found:

The *opfo-external-name* of *domain-object*.

The text returned by the function defined in the G2 parameter *devu-user-name-as-string*, if one is defined. This allows you to define an alternate method to identify domain objects. You can define a custom *devu-user-name-as-string* function by using an initialization.

The G2 name assigned to *domain-object* as text, if a name is assigned. It is not recommended that you use G2 names for identifying objects.

Text describing *domain-object*. For example:

“the item of <*class*> located on the workspace <*Workspace*>”

If you define a custom function in the G2 parameter *devu-user-name-as-string*, you must also be sure to define a new procedure to lookup an object when the identification text is passed to [devu-domain-object-lookup](#) (*object-name*: text) -> *domain-object*: class [[opfo-managed-object](#) | [opfo-containment-object](#)]. The G2 parameter *devu-alternate-object-lookup-procedure* provides the name of the alternate procedure. See [devu-alternate-object-lookup-procedure](#).

is-a-message

(*test-item*: class item)
-> *is-message*: truth-value

Returns true if *test-item* is a message or message subclass; else returns false. This function can be passed in as the *ok-to-traverse* argument to the procedures [net-traversal](#) (*node*: class item, *child-finder*: class procedure, *ok-to-traverse*: class function-definition, *node-visit-proc*: class procedure, *search-type*: symbol, *mgr*: class item, *visited-node-list*: class [devu-item-list-without-duplicates](#)) and [net-traversal-and-collect](#) (*node*: class item, *child-finder*: class procedure, *ok-to-traverse*: class function-definition, *node-visit-proc*: class procedure, *search-type*: symbol, *mgr*: class item, *list*: class [devu-](#)

[item-list-without-duplicates, bag: class item](#)) to make sure that only messages are added to the list of traversed items.

is-an-object

(*test-item*: class item)
-> *is-object*: truth-value

Returns true if *test-item* is an object or an object subclass; else returns false. This function can be passed in as the *ok-to-traverse* argument to the procedures [net-traversal](#) (*node*: class item, *child-finder*: class procedure, *ok-to-traverse*: class function-definition, *node-visit-proc*: class procedure, *search-type*: symbol, *mgr*: class item, *visited-node-list*: class devu-item-list-without-duplicates) and [net-traversal-and-collect](#) (*node*: class item, *child-finder*: class procedure, *ok-to-traverse*: class function-definition, *node-visit-proc*: class procedure, *search-type*: symbol, *mgr*: class item, *list*: class devu-item-list-without-duplicates, *bag*: class item) to make sure that only objects are added to the list of traversed items.

Methods Listed by Class

opfom-set-alarm-status

(*domain-object*: class [opfom-managed-object | opfo-containment-object],
priority: integer)

Sets the attribute `_opfom-highest-message-priority` of the domain object to *priority* and displays the alarm on the object. The object must have an alarm-region in its icon definition. For information about defining the alarm region, see the “Creating Icons for Domain Object Classes” section in the *Integrity User’s Guide*.

opfom-set-acknowledgement-status

(*domain-object*: class [opfom-managed-object | opfo-containment-object],
ack-status: symbol)

Sets attribute `_opfom-acknowledgment-status` of the domain-object to *ack-status* and displays the acknowledgment status alarm on the object. The object must have an acknowledgment-region in its icon definition. For information about defining the acknowledgment region, see the “Creating Icons for Domain Object Classes” section of the *Integrity User’s Guide*.

opfom-get-superior

(*domain-object*: class [opfom-managed-object | opfo-containment-object])
-> *superior-object*: class [opfom-managed-object | opfo-containment-object]

Returns the superior domain object of a domain object. The default superior object is the object whose subworkspace contains the domain object.

opfom-get-priority

(*domain-object*: class [opfom-managed-object | opfo-containment-object])
-> *priority*: integer

Returns the value of the attribute `_opfom-highest-message-priority` of a domain object. This attribute tells the highest priority message against the object.

opfom-get-acknowledgement-status

(*domain-object*: class [opfom-managed-object | opfo-containment-object])
-> *ack-status*: symbol

Returns the value of the attribute `_opfom-acknowledgement-status` of a domain object. The value of this attribute tells if any unacknowledged messages are targeting this object.

OPAC APIs

Describes the Integrity OPAC APIs, including object and procedure assignments, procedures called from OPAC, and additional procedures that are not called from an OPAC block.

Introduction	533
External APIs Calling OPAC from G2	534
Internal APIs for User-Written Blocks	534
Other Utility API's for User-Written Blocks	537
OPAC Error Handling	538
State Transition Diagram APIs	538
Debugging OPAC Procedures	540



Introduction

The procedures referenced inside OPAC blocks perform the following functions:

- **Control Procedure** - Determines the execution sequence for blocks that make up an OPAC graphical procedure.
- **G2-Action Procedure** - Performs the actions for a given OPAC block;
- **Decision Procedure** - Converts the decision specification to a numbered branch choice for the block.

The Control procedure should not be altered. G2 Action and Decision procedures that are user-specified are identified as such in the Configure dialog box for individual OPAC blocks.

For example, the OPAC Generic Put Something On Stack block and the User Defined Decision Procedure blocks are configured through the User Defined Decision Procedure field of the block Configure dialog window. These blocks are located on the OPAC Decisions Palette.

External APIs Calling OPAC from G2

`opac-programmed-start`

*(block: class opac-syntax-element, caller: class item, target: class object;
window: class item, notify: class object)*
-> token: class opac-token

This procedure creates a new Token, sets up the necessary relationship between the Token and its associated data, starts the OPAC procedure, then returns the Token to the calling procedure.

`opac-start-task`

*(block: class opac-syntax-element, caller: class item, target: class object;
window: class item, notify: class object)*
-> token: class opac-token

This procedure does the same as the `opac-programmed-start` procedure, but it also accepts an argument list, `arg-list`. This list is used to pass information to the OPAC procedure. Each item in the `arg-list` will be assigned to a local parameter.

Internal APIs for User-Written Blocks

The following procedures are not directly called from any OPAC block. OPAC users can create their own blocks and use the following procedures to attach to the blocks. Notice the last two arguments, `symbol` and `text`, are designated for returning an error code and an error message. An error returned by one of these procedures is signified by something other than the symbol OK.

`opac-pop-general-stack`

(token: class opac-token)
-> error-code: symbol, error-message: text

Removes the top item from the stack. This does not delete the item.

opac-task-kill-new-duplicate*(token: class opac-token)**-> error-code: symbol, error-message: text)*

If another OPAC token already exists with the same OPAC procedure, *target*, and *caller*, delete this new one and keep the old one.

opac-task-kill-old-duplicate*(token: class opac-token)**-> error-code: symbol, error-message: text)*

If another OPAC token already exists with the same OPAC procedure, *target*, and *caller*, delete the old one and keep the new one.

opac-show-stack-top-from-window*(token: class opac-token, showwindow: class g2-window)**-> error-code: symbol, error-message: text*

This procedure shows the top of the stack for the given Token. It is mainly used for testing, because in normal operation a G2 window might not be available for display, so it just picks one.

opac-show-token-info-from-window*(token: class opac-token, show-window: class g2-window)**-> error-code: symbol, errormessage: text*

Same as *opac-show-stack-top-from-window*, but displays the Token information, which includes local variables for the Token.

opac-if-token-error-free*(token: class opac-token)**-> priority: integer, errorcode: symbol, error-message: text*

Returns 1 if the highest-message-priority of *token* is greater than 99998; otherwise it returns 2. The highest-message-priority of *token* can be set by the *opac-token-error-handler*, which sets this to a value of 2. This procedure is currently used as a decision procedure and is available under the Decisions Palette within OPAC.

opac-get-local-text-var*(token: class opac-token, parm-name: text)**-> local-text-value: text, errorcode: symbol, error-message: text*

Returns the value as text for the given local parameter name.

opac-set-local-text-var*(token: class opac-token, parm-name: text, value: text)**-> error-code: symbol, error-message: text*

Sets the value of the local parameter named by *parm-name* to *value*.

opac-get-text-from-stack

(*token*: class opac-token)

-> *value-from-stack*: text, *error-code*: symbol, *error-message*: text

Retrieves the text value of the top item on the stack. This also removes the item from the stack.

opac-get-local-integer-var

(*token*: class opac-token, *parm-name*: text)

-> *local-intege-rvalue*: integer, *error-code*: symbol, *error-message*: text

Returns the integer value for the local parameter named by *parm-name*.

opac-set-local-integer-var

(*token*: class opac-token, *parm-name*: text, *value*: integer)

-> *error-code*: symbol, *error-message*: text

Sets the value of the local parameter named by *parm-name* to *value*.

opac-get-integer-from-stack

(*token*: class opac-token)

-> *value-from-stack*: text, *error-code*: symbol, *error-message*: text

Retrieves the integer value of the top item on the stack. This also removes the item from the stack.

opac-get-local-float-var

(*token*: class opac-token, *parm-name*: text)

-> *local-float-value*: float, *error-code*: symbol, *error-message*: text

Returns the float value for the local parameter named by *parm-name*.

opac-set-local-float-var

(*token*: class opac-token, *parmname*: text, *value*: float)

-> *error-code*: symbol, *error-message*: text

Sets the value of the local parameter named by *parm-name* to *value*.

opac-get-float-from-stack

(*token*: class opac-token)

-> *value-from-stack*: text, *error-code*: symbol, *error-message*: text

Retrieves the float value of the top item on the stack. This also removes the item from the stack.

opac-get-local-parameters-as-string

(*token*: class opac-token)

-> *local-parms*: text, *error-code*: symbol, *error-message*: text

Retrieves all local variables and their values, each one on a separate line. This is mainly intended for debugging purposes.

opac-get-item-via-text*(token: class opac-token, text: text)**-> item: class item, error-code: symbol, error-message: text*

Returns an item based on indirect reference through *text*. This is typically for substitution, such as, returning the item represented by, **\$caller**, or a **\$reference** to a local parameter, or also for indirect reference by text or by a text parameter. The text parameter is assumed to contain the desired reference - in that case, a recursive call of this procedure is made. The leading (\$) must be present as the first character for substitution to work.

This procedure assumes a single return, not a wildcard, which should be checked before calling this procedure. If no item is found, the Token itself is returned, and an error code is set. If *Text = \$stack*, the stack item is not consumed - it is left on the stack. It is the job of the calling program to decide what to do with the stack, or any other item returned by this procedure.

opac-token-delete*(token: class opac-token)*

Deletes all items in the stack, all local parameters, and the token itself.

Other Utility API's for User-Written Blocks

opac-get-symbol-from-text*(text: text, position-k1: integer)**-> text: text, end-position: integer*

This G2 procedure gets the next (symbol) from text *text* (in the usual parsing sense, not a G2 symbol), starting at *position-k1* in the text. Note that this returns a text. It also returns the end position of the symbol as an integer, so that you can step your way through text one symbol at a time.

This procedure looks for characters that might terminate a symbol. This includes the obvious case of blank or white space, and also includes:

{, }, (,), [,], /, +, =, !, *, ^, <, >, ?, |, :, ;, and the comma.

It uses a standard predicate *mib-character-symbol-p* to do this. Since MIB procedures must allow a (dot) in the middle of a symbol, they do not recognize periods as symbol terminators. Hence, this procedure also checks for the period.

Symbols starting with (\$) or other funny characters are recognized as full symbols, such as **\$stack**. Funny characters otherwise terminate the symbol.

opac-replace-local-parms-in-text

(*token*: class opac-token, *text*: text)

-> *actual-name*: text, *error-code*: symbol, *error-message*: text

This procedure is typically called from within OPAC blocks, so that users can use substitution variables.

This G2 procedure replaces substitution variables such as \$sender, \$target, and local variables names (also prefaced with a (\$), with their actual names. *text* refers to the text that contains the substitution variables. The first return argument contains the result of the substitution. The remaining return arguments, symbol and text are the standard error-name and error-symbol. The token is needed as input so that local names can be found.

OPAC Error Handling

opac-token-error handler

(*token*: class opac-token, *error*: symbol, *error-string*: text)

This procedure sets the token in an error state. It gathers information about the token and a traceback to the Start block for the token (see [Show Token Info](#) and [Show Stack Top](#)). This information is compiled into a message and is sent to the `devu-error-handler` (see the *Integrity User's Guide* for additional information). The If Token Error Free block can be used to test whether the token is in an error state or not (see [Special Instance: If Token Error Free Block](#)).

The `opac-token-error-handler` procedure is used to safely return from an error condition. An error condition can arise from a G2 procedure exception or any developer defined error condition. The G2 procedures used in OPAC will return the OK symbol if the procedure completed successfully and any other symbol if an error was detected (see the *G2 Reference Manual* for additional information on error conditions and error handling).

State Transition Diagram APIs

opfo-domain-object::opac-accept-new-state

(*target*: class opfo-domain-object, *new-state*: text)

-> *wait-state-block*: class item, *error-name*: symbol, *error-text*: text

This G2 procedure sets the state transition model associated with *target* to a new state specified by the text, *new-state*. If an `opac-wait-state` block that represents *new-state* exists on the same workspace associated with *target*, the current state will become *new-state* and the `opac-wait-state` block will be returned to the caller. If no `opac-wait-state` block exists, `devu-safe-object` will be returned. *new-state* may contain wildcards (* and ?) characters. This allows for partial matches to be made on the comparison of *new-state* and states

defined in the state diagram model. Note that no transition event will be generated by the result of this method.

opfo-domain-object::opac-accept-event

(*target*: class opfo-domain-object, *new-event*: text, *timeout*: integer)

-> *error-name*: symbol, *error-text*: text

This G2 procedure transitions the state transition model from one state to another state based on *new-event*. Proceeding after the *timeout* period, the current state is obtained by accessing the opac-token associated with *target*. The opac-wait-state block is then obtained by the opac-token. It then looks at each opac-transition-event block connected downstream of the opac-wait-state trying to match the *new-event* with the opac-transition-event-name of the event block. If a match is found, the procedure specified in the opac-event-action-proc of the event block is executed and the opac-token is moved to the new state. If the event block is an opac-timeout-transition-event block, the opac-token will wait at the timeout block for the time specified in opac-transition-event-timeout of the timeout block before continuing to the next state, if any. If no match is made, the opac-token remains at the current state. *new-event* may contain wildcards (* and ?) characters. This allows for partial matches to be made on the comparison of *new-state* and states defined in the state diagram model. Note that no transition event will be generated by the result of this method.

opfo-domain-object::opac-get-state

(*target*: class opfo-domain-object, *std*: symbol)

-> *current-state*: text, *error-name*: symbol, *error-text*: text

This G2 procedure retrieves the current state for the given *target*. The target must relate to an opac-token that is currently running a state transition model specified by *std*. *std* is the name of the opac-state-diagram-start block.

opfo-domain-object::opac-delete-state-token

(*target*: class opfo-domain-object, *std*: symbol)

-> *error-name*: symbol, *error-text*: text

This G2 procedure deletes the opac-token for the specified state transition model (*std*) associated with *target*.

Debugging OPAC Procedures

Blocks commonly used for debugging include:

- [Connection Post](#)
- [Show Token Info](#)
- [Put Text On Stack](#)

Note The token has a user menu choice for showing token info and showing the stack top. Normally, you pause the OPAC procedure in order to access the token user menu choices or to display its table.

Startup Parameters

Describes the startup parameters for Integrity blocks.

Introduction	541
GNDO Module	542
GMIB Module	546
GSNMP Module	546
Global Parameters	547
Performance Parameters	548



Introduction

This chapter describes the system parameters defined by using initialization items. You can change the value of these system parameters by setting the value within a GFR startup procedure. See the *G2 Foundation Resources User's Guide*.

GNDO Module

Error Handling:

These parameters define the behavior of the error handler.

`devu-alternate-object-lookup-procedure=`

Defines a procedure that looks up domain objects based on criteria other than the `opfo-external-name`. For example, if you define a class of object with the attribute `ip-address`, you can write a procedure that returns a domain object based on the value of that attribute.

The procedure you define using this initialization is called by the Integrity procedure `devu-domain-object-lookup`. The argument passed to `devu-domain-object-lookup` is passed to your new procedure. The procedure must have the following syntax:

```
custom-lookup-procedure
  (lookup-text: text)
  -> lookup-object: class: opfo-domain-object
```

Your custom lookup procedure should return the domain object that matches the text string passed to it. When you define a new procedure, it supersedes `devu-domain-object-lookup`.

`devu-error-handler-proc=smh-send-error-message`

Defines a procedure to replace the default Integrity error handler. The procedure must have the following syntax:

```
custom-error-handler
  (target: class item, sender: class item, error-type: text, priority: integer,
   error-name: text, error-text: text, error-lifetime: integer)
```

`devu-error-lifetime=300`

The number of seconds an error message is maintained before being deleted.

`devu-high-priority=1`

When the error handler generates error messages, it decides the priority of the error message by using the parameters `devu-high-priority`, `devu-medium-priority`, or `devu-low-priority`. Use this parameter to define the number assigned to a `devu-high-priority` message.

`devu-low-priority=10`

When the error handler generates error messages, it decides the priority of the error message by using the parameters `devu-high-priority`, `devu-medium-priority`, or `devu-low-priority`. Use this parameter to define the number assigned to a `devu-low-priority` message.

`devu-medium-priority=5`

When the error handler generates error messages, it specifies the priority of the error message by using the parameters `devu-high-priority`, `devu-medium-priority`, or `devu-low-priority`. Use this parameter to define the number assigned to a `devu-medium-priority` message.

`devu-status-category="Fault Expert Status"`

Specifies the text that describes any error message generated using the category `devu-status-category`. This is the default category used to generate status messages.

`devu-system-category="Fault Expert System"`

Specifies the text that describes any error message generated using the category `devu-system-category`. This is the default category used to generate system messages.

Object Retrieval

These parameters allow you to define custom procedures for retrieving an object by using an identification string. See `devu-ext-name-or-name-as-string` and `devu-domain-object-lookup`.

`devu-unknown-object-procedure=devu-default-object-return`

Defines the procedure called when no object can be found by `devu-domain-object-lookup`. The procedure you define must accept the argument shown in the following example:

```
custom-unknown-object-procedure
  (lookup-text: text)
  -> lookup-object: class: object
```

`devu-user-name-as-string=`

Defines an alternate function to retrieve a text identification from a domain object. This lets you use attributes other than `opfo-external-name` to identify an object. The function is called by `devu-ext-name-or-name-as-string`.

If you define a new function to look up the name of an object you should also define a new object lookup procedure, using `devu-alternate-object-lookup-procedure`.

The function you define must accept the argument shown in the following example:

```
custom-user-name-as-string
  (object: item)
  -> item-identifier: text
```

The function must return the text string that you want to use to identify the object passed to the function.

If you define a new function to retrieve an identifier for an object other than the `opfo-external-name`, you should also define a new procedure to lookup the object using the new identifier as described in `devu-alternate-object-lookup-procedure`.

`devui-default-workspace-for-new=`

When the system initialization `devui-use-application-object-workspaces` is set to `false`, new objects created using the object manager are not automatically placed on the application workspaces defined in an Integrity application. In this case, whenever you select Create on the object manager, a dialog appears to prompt you for the name of the workspace on which to place the new object. `devui-default-workspace-for-new` contains the name of a workspace provided as a default in that prompt. See “Objects Created in a New Application” in the *Integrity User’s Guide*.

`devui-use-application-object-workspaces=true`

When this initialization is set to `true`, new objects created using the Object Manager are placed on the application workspaces created in an Integrity application. You can set `devui-use-application-object-workspaces` to `false` to override the use of the application workspaces. When you create a new object the system prompts you for the name of the workspace on which to place the new object. Use the initialization `devui-default-workspace-for-new` to provide a default value for the prompt. See “Objects Created in a New Application” in the *Integrity User’s Guide*.

`opcsui-domain-map-workspace=unspecified`

Use this parameter if the top level of your domain map is not on the workspace created for the top level of the domain map by the `startup` module. Define the name of the workspace you used in this parameter. This will let the menu item View > Domain Map know where to retrieve the domain map.

Colors and Priority

These parameters define default colors.

`opfom-acknowledge-color=black`

Defines the color of the acknowledgment region of a domain object not targeted by any unacknowledged messages. See “Setting Priority and Acknowledgment Colors” in the *Integrity User’s Guide*.

`opfom-default-initial-priority=99999`

Defines the default value for the initial priority of the domain objects. This value is saved in the domain object attribute `_opfo-highest-message-priority`. See “Setting Default Message Priorities” in the *Integrity User’s Guide*.

opfom-default-priority-color=sky-blue

Defines the color used as a default for a message assigned a priority that has not been assigned a color using **opfom-priority-alarm-colors**. For example, a message with a priority of 15 is not mapped by the 10 color array specified as a default. Use the default color to display the message on the browser and to color the icon alarm region of the target object on the domain map. See “Setting Priority and Acknowledgment Colors” in the *Integrity User’s Guide*.

opfom-default-priority-on-delete-of-last-message=6

Defines the priority assigned to a domain object after the last message targeting it is deleted. This is stored in the object attribute **opfo-highest-message-priority**. This value can be overridden when the initialization of **opfom-revert-priority-to-initial** is set to **true**. See “Setting Default Message Priorities” in the *Integrity User’s Guide*.

opfom-priority-alarm-colors=red, orange, yellow, green, salmon, thistle, wheat, sienna, tan, sky-blue

This is a vector initialization item that defines the colors matched to message priorities. The first color in the list defines priority 1, the second defines a priority of 2, and so on. Use the color that matches a message priority to define the color used for the background of the message in the message browser. Use the item also to color the icon alarm region when the message is the highest priority message targeting the object. See “Setting Default Message Priorities” in the *Integrity User’s Guide*.

opfom-priority-alarm-text-colors=black, black, black, black, black, black, black, black, black, black

This is a vector initialization item that defines the text colors matched to message priorities. The first color in the list defines priority 1, the second defines a priority of 2, and so on. Use the color that matches a message priority to define the color used for the text of the message in the message browser. See “Setting Priority and Acknowledgment Colors” in the *Integrity User’s Guide*.

opfom-revert-priority-to-initial=true

Defines the priority assigned to a domain object after the last message targeting it is deleted. This is stored in the object attribute **_opfo-highest-message-priority**. When this item is set to **true**, the value of **_opfo-highest-message-priority** is set to the value defined by **opfom-default-initial-priority**. This initialization overrides any value set by **opfom-default-priority-on-delete-of-last-message**. See “Setting Default Message Priorities” in the *Integrity User’s Guide*.

opfom-unacknowledged-color=yellow

Defines the color used to display the icon acknowledgment region of a domain object targeted by an unacknowledged message. See “Setting Priority and

Acknowledgment Colors” in the *Integrity User’s Guide*.

GMIB Module

`mib-create-nonexistent-traps=0`

An integer parameter for specifying whether or not Integrity automatically creates trap class definitions for undefined traps. A value of 0 tells Integrity to NOT create trap class definitions and a value of 1 means do create trap class definitions.

`mib-debug=0`

An integer parameter for specifying the level of information that the Integrity knowledge base will inform the developer about. The values of this parameter are 0 - 5 (0 - No Information; 5 - Maximum Information).

`mib-receiver-time-to-live-in-queues=600`

An integer parameter, in seconds, for specifying the amount of time after which Integrity will automatically clear the `mib-reception-queue` and `mib-completed-receives` queues.

GSNMP Module

These items let you define your own procedures for handling how the `oxs_sim-request-handler` performs SNMP (SET, GET, GET NEXT) transactions.

`oxs_sim-snmp-event-gsi-obj-symbol=`

A symbolic parameter for specifying the asynchronous G2 Standard Interface object used by the G2-SNMP Bridge for receiving traps into Integrity.

`oxs_sim-snmp-get-symbol=snmp-get-rpc-call`

A symbolic parameter for specifying the procedure that the `oxs_sim-request-handler` uses in performing an SNMP GET transaction.

`oxs_sim-snmp-get-table-column-symbol=snmp-get-table-column-rpc-call`

A symbolic parameter for specifying the procedure that the `oxs_sim-request-handler` uses in performing an SNMP GET TABLE COLUMN transaction.

`oxs_sim-snmp-set-symbol=snmp-set-rpc-call`

A symbolic parameter for specifying the procedure that the `oxs_sim-request-handler` uses in performing an SNMP SET transaction.

`oxs_sim-snmp-sync-gsi-obj-symbol=g2`

A symbolic parameter for specifying the synchronous G2 Standard Interface object used by the G2-SNMP Bridge for the SNMP SET, GET, and GET NEXT

transactions. The procedure `oxs_sim-request-handler` uses this parameter in making remote procedure calls to the SNMP Gateway Bridge.

`oxs-default-unrecognized-trap-class=sample-unknown-trap-class`

A symbolic parameter used for specifying the superior class of the trap class definition that is automatically created in response to Integrity receiving an undefined trap. See `mib-create-nonexistent-traps` for information on configuring Integrity to automatically create the trap class definitions.

`oxs-running-standalone-toggle=`

An `opxb-gr-toggle-client-server-button` used for specifying to Integrity if the G2 application is running with or without the SNMP Gateway Bridge connections. The `toggle-state` attribute of this button is set to a symbolic value of `on` (running standalone) or `off` (connected to a SNMP Gateway Bridge).

`oxs-unrecognized-traps-location=oxs-unrecognized-traps`

A symbolic parameter used for specifying the location of where Integrity will store trap class definitions that are automatically created for undefined traps received. See `mib-create-nonexistent-traps` for information on configuring Integrity to automatically create the trap class definitions.

Global Parameters

`asn1_integer=2`

A global integer parameter for specifying the ASN.1 integer type. The default value is 2. This value should not be modified.

`asn1_octet_string=4`

A global integer parameter for specifying the ASN.1 octet string type. The default value is 4. This value should not be modified.

`snmp_get_req_msg=160`

A global integer parameter for the SNMP GET request message type. The default value is 160. This value should not be changed.

`snmp_getnext_req_msg=161`

A global integer parameter for the SNMP GET NEXT request message type. The default value is 161. This value should not be changed.

`snmp_get_rsp_msg=162`

A global integer parameter for the SNMP GET response message type. The default value is 162. This value should not be changed.

`snmp_set_req_msg=163`

A global integer parameter for the SNMP SET message type. The default value is 163. This value should not be changed.

`snmp_trap_req_msg=164`

A global integer parameter for the SNMP Trap message type. The default value is 164. This value should not be changed.

Performance Parameters

The Integrity performance parameters are used in measuring the SNMP trap input performance. These parameters can be further used in performing various diagnostic actions in response to out of tolerance conditions.

MIB Module

`mib-number-of-reception-queue-discards`

An integer parameter giving the count of MIB-RECEIVER objects that have been deleted from the item list `mib-reception-queue`.

`mib-number-of-completed-receive-discards`

An integer parameter giving the count of MIB-RECEIVER objects that have been deleted from the `mib-completed-receives` queue.

`mib-maximum-complete-receive-queue-length`

An integer parameter giving the maximum number of MIB-RECEIVER objects in the `mib-completed-receives` queue. This value should be used in conjunction with the `mib-receiver-time-to-live-in-queues` initialization to determine if your application is getting behind in the processing of incoming traps.

`mib-number-of-expected-val-vs-receive-mismatches`

An integer parameter giving the total count of MIB-RECEIVER objects whose `Number-of-received-values` does not match its `Number-of-expected-values`.

`mib-number-of-receive-errors`

An integer parameter giving the total count of MIB-RECEIVER objects return errors based upon the `Number-of-errors` attribute of the MIB-RECEIVER object. This is *not* a count of MIB-RECEIVER objects with return errors but a count of total return errors for all MIB-RECEIVER objects.

`mib-number-of-completed-receives`

An integer parameter giving the total count of MIB-RECEIVER objects processed by Integrity.

GSNMP Module

snmp-number-of-ignored-traps

An integer parameter giving the total count of incoming traps that are not processed by the Integrity knowledge base. Ignored traps are defined as follows:

No trap class definition could be found for the trap, including TRAP-XX-XX-XX.

The integer parameter `mib-create-nonexistent-traps` equals zero. Unknown traps *do not* have a trap class definition created for them automatically. See `mib-create-nonexistent-traps` for more information.

snmp-number-of-ignored-trap-params

An integer parameter giving the total count of OID variable values that could not be matched with their respective incoming trap.

snmp-number-of-traps-created

An integer parameter giving the total count of MIB-RECEIVER objects created as a result of a trap class definition not being found in Integrity for the appropriate TRAP-[ENTERPRISE ID]-[GENERIC ID]-[SPECIFIC ID] information of the trap. The integer parameter `mib-create-nonexistent-traps` must be set to a value of 1 in order for undefined traps to be created automatically.

snmp-traps-per-second-min

A float variable giving an average of the `mib-number-of-completed-receives` over the last 10 seconds. The default update rate for this variable is none. See the *oxs_demo.kb* for an example action button for setting this value; **Get, Set & Send Traps > Performance Parameters**.

snmp-ignored-traps-per-second-min

A float variable giving an average of the `snmp-number-of-ignored-traps` over the last 10 seconds. The default update rate for this variable is none. See the *oxs_demo.kb* for an example action button for setting this value; **Get, Set & Send Traps > Performance Parameters**.

snmp-traps-per-second

A float variable giving the rate of change per second of the integer parameter `mib-number-of-completed-receives` during the last 2 seconds. The default update rate for this variable is none. See the *oxs_demo.kb* for an example action button for setting this value; **Get, Set & Send Traps > Performance Parameters**.

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

A

agent: (SNMP) Software installed in each network device; answers incoming data requests from a manager, and sends unsolicited messages to a manager. Generally does not interact directly with a human operator. Communicates with a manager using a standard protocol such as SNMP, and stores data in a standard format, e.g., as also defined by SNMP.

alarm: (1) a high priority event, including combinations of related events. (2) a visible indication, to the Operator, of a potential problem
CLASS: an object template which defines a specific set of attributes and behaviors for all instances of the class.

B

Blocking: When a call to a remote procedure in the SNMP Gateway Bridge is blocking, neither the bridge nor Integrity can make other procedure calls while the remote procedure call is being processed.

C

class: An object template which defines a specific set of attributes and behaviors for all instances of the class.

correlation: the process of grouping related items.

E

enterprise: (SNMP) an object identifier, usually for a vendor product, that names the originator of a trap. The first field of an SNMP trap.

event: a relatively instantaneous occurrence that changes the state of a system or element.

F

fault: a failure of some element of a system. It may or may not be detected.

filter: a selection process which eliminates signals or data which do not fall within specified bounds, e.g., values that are lower than a specified threshold.

I

icon: a pictorial representation of an object. In G2 icons are defined at the class level.

M

management information base(mib): a set of managed objects that hold network information, generally maintained by agents.

manager: Software which receives data or may access data from agents. The agents maintain the data, while the manager normally interacts with a human operator.

message: Represents an event, contains text and other attributes

N

non-blocking: When a call to a remote procedure in the SNMP Gateway Bridge is non-blocking, both the bridge and Integrity can make other calls while the call to the Integrity receiver procedure is being processed.

O

object: in G2, is a subclass of the built-in Item class.

P

proxy agent: (SNMP) an agent that acts as a protocol converter for access to data in non-standard MIBs, so that a manager can access the data using a standard protocol such as SNMP.

S

SGB: Simple Network Management (SNMP) Gateway Bridge. The GSI based component of the G2-SNMP Bridges.

SNMP: Simple Network Management Protocol.

state: the mode or condition of an object or system as defined by the values of its attributes.

status: SYMPTOM: a "zero cost" test. Usually observed through passive monitoring of a system.

symptom: A zero cost test. Usually observed through passive monitoring of a system.

T

test: Some action that produces a pass/fail result.

trap: (SNMP) a message that reports a problem or a significant event.

W

workspace: an entity in G2 upon which objects are placed. It loosely resembles a window, except that it may be a "subworkspace" of another object, usually indicating a containment relationship.

@ 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

Symbols

_mpe-from-message-filter
 _mpe-from-text-buffer

Numerics

2 Way Decision block
 manual
 pattern
 2-Way-Pattern-Decision block
 3 Way Decision block
 manual
 pattern
 4 Way Decision block
 manual
 pattern

A

Accept New Event block
 Acknowledge Message
 Acknowledge Message block
 acknowledge messages
 acquiring data from G2-SNMP Bridges
 Add Passport to Event
 adding a filtered trap
 agent
 hostnames
 IP addresses
 API
 functions
 devu-ext-name-or-name-as-string
 is-a-message
 is-an-object
 methods
 opfom-get-acknowledgement-status
 opfom-get-priority
 opfom-get-superior
 opfom-set-acknowledgement-status
 opfom-set-alarm-status
 opac-get-float-from-stack
 opac-get-integer-from-stack

opac-get-item-via-text
 opac-get-local-float-var
 opac-get-local-parameters-as-string
 opac-get-text-from-stack
 opac-if-token-error-free
 opac-programmed-star
 opac-set-local-float-var
 opac-set-local-integer-var
 opac-show-stack-top-from-window
 opac-show-token-info-from-window
 opac-start-task
 opfo-domain-object
 opac-accept-event
 opac-accept-new-state
 opac-delete-state-token
 opac-get-state
 procedures
 devu-consume-next-field
 devu-decode-comma-line
 devu-devu-safe-window
 devu-domain-object-lookup
 devu-error-handler
 devu-get-field-n
 devu-insert-item-in-sorted-ascending-list
 devu-insert-item-in-sorted-descending-list
 devu-insert-msg-in-alphabetical-list
 devu-insert-msg-in-alphabetical-list-allowing-duplicates
 devu-pattern-matcher-with-indices
 devu-safe-text-for-symbol
 devu-strip-linefeeds
 devu-substitute-text
 devu-text-is-ascii-punctuation-p
 devu-text-is-ascii-symbol-p
 devu-txt-is-digit-p
 devu-txt-is-mib-character-symbol-p
 do-nothing-at-node-collection-visit
 do-nothing-at-node-visit
 find-connected-children
 find-downstream-connected-children
 find-downstream-connected-children-2

- find-subworkspace-children
 - find-upstream-connected-children
 - glf-disable-logging
 - glf-enable-logging
 - glf-set-fixed-log-closing-times
 - glf-write-to-log-file
 - net-traversal
 - net-traversal-2
 - net-traversal-and-collect
 - net-traversal-and-collect-2
 - sc-toggle-button
 - smh-acknowledgement-proc
 - smh-create-message
 - smh-delete-all-history
 - smh-delete-history
 - smh-delete-message
 - smh-delete-server
 - smh-get-message-history
 - smh-get-messages-about
 - smh-get-messages-in-server
 - smh-get-messages-sent-by
 - smh-message-delete-proc
 - smh-message-query
 - smh-propagate-message-text
 - smh-read-server-mib-from-file
 - smh-send-error-message
 - smh-server-delete-all-msg
 - smh-write-server-mib-to-file
 - twm-hide-screen
 - APIs calling OPAC
 - applications
 - G2-SNMP Bridges used for
 - argument
 - argument block
 - ASCII text file
 - asn1_integer global parameter
 - asn1_octet_string global parameter
 - asn1-rfc1212-mib-parser procedure
 - assign a value
 - Attribute Filter
 - attribute, message
 - attributes of
 - GSI Interface Object
 - authorizing
 - G2-SNMP Bridges
 - SNMP Gateway Bridge (SGB)
- B**
- Beep
- C**
- beginning
 - block definitions
 - block filtering
 - blocking remote procedure calls
 - blocking SNMP transactions
 - boot process
 - branching capability
 - two-way
 - building an G2-SNMP Bridges application
- C**
- calling OPAC from G2
 - CD-ROM
 - installing G2-SNMP Bridges from
 - mounting
 - unmounting
 - Clear Message History block
 - clearing mib-receiver object queues
 - clears for attribute
 - format of
 - Clears for or Delete Messages
 - collect items
 - Collect Related Items block
 - column height
 - community name string
 - community-length
 - community-name
 - Comparison Decision block
 - completion procedure determination
 - components
 - G2-SNMP Bridges
 - G2-SNMP Generic Bridge
 - configuration
 - G2-SNMP Bridges
 - G2-SNMP Bridges communication
 - parameters
 - GSI Interface Object
 - simulated SNMP traps
 - variable fields of
 - SNMP agent MIB
 - tables of
 - variables of
 - connect
 - connected-downstream-to
 - connected-to
 - connected-upstream-to
 - Connection Posts
 - Connection-Post
 - Connection-Post block

- connectivity
- Consume Decision From Stack
- Control Delay block
- Control-Procedure
- CORE Services
- Count Events
- create a message
- creating
 - a new error handling procedure
 - GSI OK file
 - trap class definitions
- Current Message Query block
- current messages
- current state
- customer support services

D

- daemon
 - trapd*
- Day of the Week Filter
- debugging
- Debugging Blocks palette
 - Show Stack Top
 - Show Token
- Debugging OPAC procedures
- debugging procedures
- Decision Blocks palette
 - 2 Way Manual Decision
 - 2 Way Pattern Decision block
 - 3 Way Manual Decision
 - 3 Way Pattern Decision Block
 - 4 Way Manual Decision
 - 4 Way Pattern Decision Block
 - Comparison Decision
 - If Token Error Free
- Decision-Procedure
- delete a file
- Delete Event
- Delete Events
- Delete Events by Start Time
- Delete File block
- Delete Message block
- delete messages
- delete old events
- deleting a filtered trap
- demonstration
 - G2-SNMP Bridges
- device name
 - determining on UNIX platform
- devu module API procedures

- devu-alternate-object-lookup-procedure
 - initialization
- devu-consume-next-field procedure
- devu-decode-comma-line procedure
- devu-domain-object-lookup procedure
- devu-error-handler procedure
- devu-error-handler-proc initialization
- devu-error-lifetime initialization
- devu-ext-name-or-name-as-string function
- devu-get-field-n procedure
- devu-high-priority initialization
- devu-insert-item-in-sorted-ascending-list
 - procedure
- devu-insert-item-in-sorted-descending-list
 - procedure
- devu-insert-msg-in-alphabetical-list procedure
- devu-insert-msg-in-alphabetical-list-allowing-duplicates procedure
- devu-low-priority initialization
- devu-medium-priority initialization
- devu-pattern-matcher-with-indices procedure
- devu-safe-text-for-symbol procedure
- devu-safe-window procedure
- devu-status-category initialization
- devu-strip-linefeeds procedure
- devu-substitute-text procedure
- devu-system-category initialization
- devu-txt-is-ascii-punctuation-p procedure
- devu-txt-is-ascii-symbol-p procedure
- devu-txt-is-digit-p procedure
- devu-txt-is-mib-character-symbol-p procedure
- devu-unknown-object-procedure initialization
- devu-user-name-as-string initialization
- domain objects
- DXIOV-IMPORT-OBJECT
- DXIOV-TYPE-TO-CLASS-OBJECT

E

- end
- Enterprise/generic/specific filtering
- environment variable
 - G2_PPD_FILENAME*
 - GSI_ROOT*
 - NEWSTATUS*
 - OBJECT*
 - setting on UNIX platform
- error
 - General failure on agent
- error handler
 - creating a new error handling procedure

- defining
- error state
- event
 - progress through G2-SNMP Bridge
- Event Action Procedure
- Event Class Filter
- Event Count by Start Time
- executing
 - G2-SNMP Bridges
 - simulated traps
 - SNMP Gateway Bridge (SGB)
- existence, message
- Export Map
- expression matching
- External Interfaces palette
 - Read Domain Map block
 - Send CDG Event
 - SNMP Get
 - SNMP Get Table Column
 - SNMP Set

F

- facilities for simulating
- file existence
- File Exists Test block
- file name for filtering log file
- Filter Block
 - (*filter_mode*)
- filtered hosts
- filtered hosts list
- filtering
 - Enterprise/generic/specific
 - hostname/IP address
 - stages
 - turning on and off
- filtering log file
- filtering traps
 - from specified hosts
- float parameter
- float parameters
- floating point value
- for loop
- four-way branching capability
- function call
- functions
 - devu-ext-name-or-name-as-string*
 - is-a-message*
 - mib-oid-strip-instance-number*
 - mib-oid-symbol-to-name*

- mib-oid-text-to-name*
- snmp-desc-value*
- snmp-severity-to-status-text-conversion*

G

- G2 Procedure Response
- G2 procedure statements
- G2 Standard Interface (GSI)
 - G2_PPD_FILENAME* environment variable
- G2-Action-Procedure
- G2-SNMP Bridges
 - acquiring data from
 - adding and deleting filtered traps
 - application block diagram
 - authorizing
 - building an application
 - completion procedure determination
 - completion procedures
 - components of
 - configuration
 - configuring communication parameters
 - creating a new error handling procedure
 - defining error handler
 - determining trap class to create
 - environment variable
 - event progress
 - executing
 - features and benefits of
 - filter definition file required
 - filtering traps
 - Gensym provided MIBs
 - GSI based component
 - handling unrecognized traps
 - installing from CD-ROM on unix
 - installing from tape on unix
 - integer value range
 - Integrity
 - intended applications of
 - modes of operation of
 - outline of setup steps
 - overloading remote procedure calls
 - performance parameters
 - port-number
 - ports for
 - processing *trapd.conf* file
 - processing *trapd.conf.ppd* file
 - relation to Integrity
 - running as a background process
 - sending traps to external systems

- setting up clears for simulation facilities
- SNMP Gateway Bridge (SGB)
- SNMP Gateway Bridge (SGB) processes
- SNMP transactions
- status messages reported by transactions
 - blocking
 - non-blocking
- trap class creation
- trap handling
 - trapd.conf* filter definition
 - trapd.conf.ppd* file parser
 - unable to find port
 - unique identifier for SNMP transactions
 - viewing unrecognized SNMP Traps
 - workspace
 - what are they ?
- G2-SNMP Generic Bridge
 - components of
- G2SNMP_ADD_FILTERED_TRAP
- G2SNMP_ADD_FILTERED_TRAP remote procedure call
- G2SNMP_BLOCKING_TRANSACTION base remote procedure call
- G2SNMP_DELETE_FILTERED_TRAP
- G2SNMP_DELETE_FILTERED_TRAP remote procedure call
- G2SNMP_MODIFY_COMM_PARAMS remote procedure call
- G2SNMP_NONBLOCKING_TRANSACTION base remote procedure call
- G2SNMP_NONBLOCKING_TRANSACTION base remote procedure call
- G2SNMP_RECEIVE_EOT receiver procedure
- G2SNMP_RECEIVE_FLOAT receiver procedure
- G2SNMP_RECEIVE_INTEGER receiver procedure
- G2SNMP_RECEIVE_MESSAGE receiver procedure
- G2SNMP_RECEIVE_STRING receiver procedure
- G2SNMP_RECEIVE_TRAP_PACKET receiver procedure
- G2SNMP_SET_AGENT_FILTER_MODE
- G2SNMP_USE_SNMP_COMM_PARAMS remote procedure call
- G2SNMP_USE_SNMP_DEFAULTS remote procedure call
- Gather Evidence
- General Actions Palette
 - Connection Posts
 - Control Delay block
 - General Procedure block
 - Hide Workspace block
 - Historical Query block
 - Macro block
 - Pause Capability block
 - Procedure Statement block
 - Procedure Template
 - Send SMH Message block
 - Show Workspace block
 - Show Workspace Not Stacked block
 - Subtask block
 - Subtask Completion block
 - Subtask Start block
 - Task Kill block
 - Task Spawn block
- General Procedure block
- Generic Blocks palette
 - Collect Related Items
 - Iteration
 - Numerical Query with Threshold
 - Run Domain Object Method
 - Set Local Parameter from Source
- Generic Bridge
 - components of
 - straps* process
- Generic Put Something on Stack Block
- Get Event Attribute
- Get State block
- GET_2_BLOCKING overloaded remote procedure call
- GET_BLOCKING_SINGLE overloaded remote procedure call
- GET_NONBLOCKING_SINGLE overloaded remote procedure call
- gethostbyname()
- glf module API procedures
- glob-style regular expression matching
- GSI Interface Object
 - connecting G2 to GSI
 - updating attributes
- GSI OK file
 - creating
 - example
 - format
- GSI_ROOT* environment variable
- GSI-based Bridge Process

H

- Hide Workspace
- Hide Workspace block
- Historical Message Query block
- Historical-Message-Query block
- host filtering mode
- hostname
- hostname/IP address filtering
- hostnames
 - agent
- hosts
 - filtered
 - filtered list
 - specified
- Hour of the Day Filter
- HP OpenView
 - sending a status trap to

I

- If Token Error Free block
- IF-TOKEN-ERROR-FREE block
- Import Map option
- Indirect References
- initalizations
 - user-defined-error-handler
- initializations
 - by groupings
 - error handling
 - object retrieval
 - SNMP transactions
 - devu-alternate-object-lookup-procedure
 - devu-error-handler-proc
 - devu-error-lifetime
 - devu-high-priority
 - devui-default-workspace-for-new
 - devui-use-application-object-workspaces
 - devu-low-priority
 - devu-medium-priority
 - devu-status-category
 - devu-system-category
 - devu-unknown-object-procedure
 - devu-user-name-as-string
 - mib-create-nonexistent-trap
 - mib-create-nonexistent-traps
 - mib-debug
 - mib-receiver-time-to-live-in-queues
 - opcsui-domain-map-workspace
 - opfom-acknowledge-color
 - opfom-default-initial-priority

- opfom-default-priority-color
- opfom-default-priority-on-delete-of-last-message
- opfom-priority-alarm-colors
- opfom-priority-alarm-text-colors
- opfom-revert-priority-to-initial
- opfom-unacknowledged-color
- oxs_sim-snmp-event-gsi-obj-symbol
- oxs_sim-snmp-get-symbol
- oxs_sim-snmp-get-table-column-symbol
- oxs_sim-snmp-set-symbol
- oxs_sim-snmp-sync-gsi-obj-symbol
- oxs-default-unrecognized-trap-class
- oxs-running-standalone-toggle
- oxs-unrecognized-traps-location
- install script
 - unix platform
- installing
 - from CD-ROM on unix
 - G2-SNMP Bridges
 - from tape on unix
 - G2-SNMP Bridges
 - OV Map Importer
- integer parameter
- integer parameters
- integer value
- integer values
 - range of
- Integrity
 - CORE Services
 - defining error handler
 - demonstration of
 - G2-SNMP Bridges
 - OPAC
- IP address
- IP address filtering
- IP addresses
 - agent
- IP Reachability Analyzer
 - See IPRA
- ipra-ora-two-report
- is-a-message function
- Item Argument block
- Iteration block
- iterations

K

- kill a process
- Kill Process block

L

- laying out objects
- list of messages
- Local Float Variable block
- local integer parameter
- Local Integer Parameter block
- local integer parameters
- Local Item block
- local items
- Local Parameter
- local parameter
- Local Parameters palette
 - Local Float Variable
 - Local Integer Parameter
 - Local Item
 - Local Text Parameter
 - Set Local Float from Source
 - Set Local Integer from Source
 - Set Local Item from Source
 - Set Local Text from Source
- Local Text Parameter block
- local text parameters
- Log_Event

M

- macro
- Macro block
- Management Information Base (MIB)
 - provided with G2-SNMP Bridges
 - simulation facilities
 - vendor supplied
- message
- message attribute
- Message block
- Message Exists block
- message filter attributes
 - Mpe contains expression
 - Mpe end expression
 - Mpe start expression
- Message Historical Query Filter
- message histories
- Message palette
 - Acknowledge Message
 - Clear Message History
 - Current Message Query
 - Delete Message
 - Message Exists
 - Send SMH Message block
 - Set Message Attribute

- Message Query Filter
- message server
- messages
- messages, delete
- messages,acknowledge
- messages,current
- mib-clean-up-ws procedure
- mib-create-name-to-oid-translation procedure
- mib-create-nonexistent-trap initialization
- mib-create-nonexistent-traps initialization
- mib-debug initialization
- mib-delete-dictionary-on-workspace procedure
- mib-delete-old-list-entries procedure
- mib-enterprise-oid-text-to-name procedure
- mib-maximum-complete-receive-queue-length
 - performance parameter
- mib-number-of-completed-receive-discards
 - performance parameter
- mib-number-of-completed-receives
 - performance parameter
- mib-number-of-expected-val-vs-receive-mismatches
 - performance parameter
- mib-number-of-receive-errors
 - performance parameter
- mib-number-of-reception-queue-discards
 - performance parameter
- mib-oid-strip-instance-number function
- mib-oid-symbol-to-name function
- mib-oid-text-to-name function
- mib-read-clears-file procedure
- mib-receiver object
 - clearing queues
 - queues
- mib-receiver-create-and-queue procedure
- mib-receiver-time-to-live-in-queues
 - initialization
- mib-return-default-msg-category procedure
- mib-substitute-in-format-spec procedure
- mib-translate-info-to-completion-name
 - procedure
- mib-translate-info-to-trap-name procedure
- mib-trapd-preprocessed-conf-reader
 - procedure
- mib-write-clears-file procedure
- modes of operation of G2-SNMP Bridges
- mount
 - CD-ROM
- Mpe contains expression attribut
- Mpe end expression attribute
- MPE Palette Blocks
- Mpe start expression attribute

mpe-add-text-to-buffer
 mpe-clear-buffer
 mpe-contains-expression
 mpe-current-real-time-as-time-stamp
 mpe-data-source-description
 mpe-data-source-name
 mpe-debug-on
 mpe-delay
 mpe-description
 mpe-destination
 mpe-end-expression
 mpe-end-position
 mpe-end-regex-expression
 mpe-extract-mode
 mpe-filter-rejects-buffer-max-entries
 mpe-filter-rejects-to-bottom
 mpe-filter-rejects-to-filter-rejects-buffer
 mpe-from-message-filter
 mpe-from-text-buffer
 mpe-line-delimiter
 mpe-line-number
 mpe-match-end
 mpe-match-source
 mpe-match-start
 mpe-match-string
 mpe-maximum-buffer-length
 mpe-message-filter
 mpe-message-filter block
 mpe-pause-block
 mpe-procedure-conclude-block
 mpe-procedure-name
 mpe-receiver-text
 mpe-regex-expression
 mpe-show-buffer
 mpe-single-match-decision-block
 mpe-single-regex-conclude
 mpe-single-regex-conclude-block
 mpe-start-end-match-decision-block
 mpe-start-end-of-match-decision-block
 mpe-start-end-of-text-conclude-block
 mpe-start-end-regex-conclude-block
 mpe-start-expression
 mpe-start-of-text-to-end-of-regex-conclude-
 block
 mpe-start-of-text-to-end-regex-conclude-block
 mpe-start-position
 mpe-start-regex-expression
 mpe-start-search-position
 mpe-static-conclude-block
 mpe-string-position-block
 mpe-string-receiver

mpe-string-receiver-block
 mpe-string-receiver-class
 mpe-terminal-block
 mpe-text-buffer
 mpe-text-source
 mpe-turn-debugging-off
 mpe-turn-debugging-on
 mpe-value
 mpe-word-delimiter
 mpe-word-line-block
 mpe-word-line-conclude-block
 mpe-word-number
 mpe-working-text
 mpe-write-mode

N

new state
 NewEvent
 non-blocking remote procedure calls
 non-blocking SNMP transactions
 Numerical Query with Threshold block
 numerical values
 numerical variables

O

Object ID
 Object Identifier
 defined
 ocal parameter
 Odie Events
 ODIE Filters
 ODIE Managers
 ODIE Responses
 ODIE Subscribers
 ODIE Wildcard'
 odie-event
 odie-event-proxy
 odie-g2-manager
 odie-g2-manager::odie-datastore-add-event-
 passport-stamp
 odie-g2-manager::odie-datastore-delete-event
 odie-g2-manager::odie-datastore-delete-events
 odie-g2-manager::odie-datastore-duration-
 count-query
 odie-g2-manager::odie-datastore-duration-
 proxy-query
 odie-g2-manager::odie-datastore-get-event-
 attribute-value

- odie-g2-manager::odie-datastore-get-passport-stamps
- odie-g2-manager::odie-datastore-set-event-attribute-value
- odie-g2-manager::odie-datastore-start-time-count-query
- odie-g2-manager::odie-datastore-start-time-proxy-query
- odie-manager::odie-manager-add-event-passport-stamp
- odie-manager::odie-manager-create-event-class
- odie-manager::odie-manager-delete-event
- odie-manager::odie-manager-delete-events
- odie-manager::odie-manager-duration-count-query
- odie-manager::odie-manager-duration-proxy-query
- odie-manager::odie-manager-get-event-attribute-value
- odie-manager::odie-manager-get-passport-stamps
- odie-manager::odie-manager-post-inform-statement
- odie-manager::odie-manager-publish-existing-event
- odie-manager::odie-manager-publish-new-event
- odie-manager::odie-manager-set-event-attribute
- odie-manager::odie-manager-start-time-proxy-query
- odie-manager::odie-manager-subscribe-event-class
- odie-manager::odie-manager-substitute-attribute-values
- odie-manager::odie-manager-unsubscribe-event-class
- OID
 - defined
- ompe-additional-text
- Ompe-additional-text attribute
- ompe-category
- Ompe-category attribute
- ompe-category-to-delete
- ompe-create-message-block
- ompe-delete-message block
- ompe-delete-message-block
- ompe-go-to-procedure
- ompe-lifetime
- ompe-local-arguments
- ompe-message-server
- ompe-message-text
- Ompe-message-text attribute
- ompe-opac-subtask-start-block
- ompe-opac-subtask-start-name
- ompe-options
- ompe-priority
- ompe-sender
- ompe-string-receiver
- ompe-target
- OPAC
 - error handling procedure
 - miscellaneous procedures
- OPAC Agent Hostname
- OPAC blocks
 - debugging
 - decision
 - external interfaces
 - general actions
 - generic
 - local parameters
 - state transition
 - subtask arguments
 - used to build state transition models
- OPAC Blocks for ODiE Events
- OPAC Error Handling
- OPAC General Procedure
- OPAC requirements
- OPAC State Diagram Examples
- Opac State Transition palette
 - Accept New Event
 - Get State
 - Transition Event
 - Wait State
- OPAC Subtask
- OPAC-2-Way-Decision
- OPAC-2-Way-Decision block
- OPAC-2-Way-Decision-By-Symbol
- OPAC-2-Way-Manual-Decision
- OPAC-2-Way-Manual-Decision block
- OPAC-2-Way-Pattern-Decision
- OPAC-2-Way-Pattern-Decision-By-Symbol block
- OPAC-3-Way-Decision
- OPAC-3-Way-Manual-Decision
- OPAC-3-Way-Manual-Decision block
- OPAC-3-Way-Pattern-Decision
- OPAC-4-Way-Decision
- OPAC-4-Way-Manual-Decision
- OPAC-4-Way-Manual-Decision block
- OPAC-4-Way-Pattern-Decision

OPAC-4-Way-Pattern-Decision block
 OPAC-Accept-New-Event block
 OPAC-Accept-New-Event-Block
 OPAC-Accept-New-State block
 OPAC-Accept-New-State-Block
 OPAC-Acknowledge-Message
 OPAC-Acknowledge-Message block
 OPAC-Block-Pause-Capability
 OPAC-Block-Pause-Capability block
 OPAC-CLEAR-Message-History
 OPAC-Clear-Message-History block
 OPAC-Comparison-Decision
 OPAC-Comparison-Decision block
 OPAC-Control-Delay
 OPAC-Control-Delay block
 OPAC-Current-Message-Query
 OPAC-Current-Message-Query block
 OPAC-Decision-Procedure
 OPAC-Delete-File
 OPAC-Delete-File block
 OPAC-Delete-Message
 OPAC-Delete-Message block
 OPAC-Delete-State-Token block
 OPAC-Delete-State-Token-Block
 OPACDEMO.KB module
 OPAC-Domain-Object-Method
 OPAC-File-Exists-Test
 OPAC-File-Exists-Test block
 OPAC-General-Procedure
 OPAC-General-Procedure block
 OPAC-Generic-Put-Something-On-Stack
 OPAC-Generic-Put-Something-On-Stack block
 opac-get-local-text-var
 OPAC-Get-Related-Items block
 OPAC-Get-Related-Items-Block
 OPAC-Get-State block
 OPAC-Get-State-Block
 opac-get-symbol-from-text
 OPAC-Hide-Workspace block
 OPAC-Historical-Message-Query
 OPAC-Historical-Numerical-Query
 OPAC-Historical-Numerical-Query block
 OPAC-Item-Arg
 opac-item-arg
 OPAC-Item-Argument block
 OPAC-Item-Argument blocks
 OPAC-Iteration
 OPAC-Iteration block
 OPAC-Iteration-Procedure-Template
 OPAC-Kill-Process
 OPAC-Kill-Process block
 OPAC-Local-Float-Parameter
 OPAC-Local-Float-Parameter block
 OPAC-Local-Integer-Parameter
 OPAC-Local-Integer-Parameter block
 OPAC-Local-Item
 OPAC-Local-Item block
 OPAC-Local-Text-Parameter
 OPAC-Local-Text-Parameter block
 OPAC-Macro
 OPAC-Macro block
 OPAC-Message-Attribute-Procedure-
 Template
 OPAC-Message-Exist
 OPAC-Message-Exists block
 OPAC-New-Procedure
 OPAC-POP-General-Stack
 OPAC-Pop-General-Stack block
 OPAC-POP-General-Stack-And-Delete
 OPAC-Pop-General-Stack-And-Delete block
 OPAC-Procedure
 OPAC-Procedure-Statement
 OPAC-Procedure-Statement block
 OPAC-Put-Connected-ObjectS-On-Stack
 OPAC-Put-Float-On-Stack
 OPAC-Put-Float-On-Stack block
 OPAC-Put-Integer-On-Stack
 OPAC-Put-Integer-On-Stack block
 OPAC-Put-Item-On-Stack
 OPAC-Put-Item-On-Stack block
 OPAC-Put-Text-On-Stack
 OPAC-Put-Text-On-Stack block
 OPAC-Read-Domain-Map
 OPAC-Read-Domain-Map block
 OPAC-Read-File
 OPAC-Read-File block
 opac-replace-local-parms-in-text
 OPAC-Run-Domain-Object-Method
 OPAC-Run-Domain-Object-Method block
 OPAC-Send-CDG-Event
 OPAC-Send-CDG-Event block
 OPAC-Send-SMH-Message block
 OPAC-Set-Local-Float-From-Source
 OPAC-Set-Local-Float-From-Source block
 OPAC-Set-Local-Integer-From-Source
 OPAC-Set-Local-Integer-From-Source block
 OPAC-Set-Local-Item-From-Source
 OPAC-Set-Local-Item-From-Source block
 OPAC-Set-Local-Parameter-From-Source
 OPAC-Set-Local-Parameter-From-Source
 block
 OPAC-Set-Local-Text-From-Source

- OPAC-Set-Local-Text-From-Source block
- opac-set-local-text-var
- OPAC-Set-Message-Attribute
- OPAC-Set-Message-Attribute block
- OPAC-Show-Stack-Top
- OPAC-Show-Stack-Top block
- opac-show-stack-top-from-window
- OPAC-Show-Token-Info
- OPAC-Show-Token-Info block
- OPAC-Show-Workspace
- OPAC-Show-Workspace block
- OPAC-Show-Workspace-NOT-Stacked
- OPAC-SMH- Create-Message API
- OPAC-SNMP-Get
- OPAC-SNMP-Get block
- OPAC-SNMP-Get-Table-Column
- OPAC-SNMP-Get-Table-Column block
- OPAC-SNMP-Set
- OPAC-SNMP-SET block
- OPAC-Spawn-No-Return
- OPAC-Spawn-No-Return block
- OPAC-Spawn-Return-Output
- OPAC-Spawn-Return-Output block
- OPAC-Spawn-Return-PID
- OPAC-State-Diagram-Completion
- OPAC-State-Diagram-Completion block
- OPAC-State-Diagram-Start
- OPAC-State-Diagram-Start block
- OPAC-Subtask block
- OPAC-Subtask-Block
- OPAC-Subtask-Completion
- OPAC-Subtask-Completion block
- OPAC-Subtask-Start
- OPAC-Subtask-Start block
- OPAC-Task-Kill
- OPAC-Task-Kill block
- opac-task-kill-new
- OPAC-Task-Spawn
- OPAC-Task-Spawn block
- OPAC-Timeout-Transition-Event
- opac-token-delete
- opac-token-error handler
- OPAC-Transition-Event
- OPAC-Transition-Event block
- OPAC-Value-Arg
- OPAC-Value-Argument block
- OPAC-Value-Argument blocks
- OPAC-Wait-State
- OPAC-Wait-State block
- OPAC-Write-Domain-Map
- OPAC-Write-Domain-Map block
- OPAC-Write-File
- OPAC-Write-File block
- OpenView Map Importer
 - See OV Map Importer
- Operating System (OS) Actions palette
 - Delete
 - File Exists Test
 - Kill Process
 - Read File Block
 - Spawn No Return
 - Spawn Return Output
 - Spawn Return PID
 - Write File
- OpEx
 - Application Programmer? Interface (API)
- OpEx functions
- OpEx methods
- OpEx Reachability Analyzer
 - See ORA-TWO
- ora-release-object-states
- ORA-TWO
 - additional procedures
 - ora-two-get-state
 - ora-two-release-object-states
 - concepts
 - domain methods
 - ora-two-collect-non-terminal-nodes
 - ora-two-collect-passive--nodes
 - ora-two-collect-terminal-nodes
 - ora-two-domain-consistency-check
 - ora-two-node-type
 - ora-two-poll-fail-message
 - ora-two-poll-node
 - ora-two-poll-recover-method
 - ora-two-predicted-poll-fail-message
 - ora-two-recursive-collect-non-terminal-nodes
 - ora-two-recursive-collect-passive-nodes
 - ora-two-recursive-collect-terminal-nodes
 - ora-two-root-cause-message
 - event methods
 - introduction to
 - manager object
 - node types
 - polling
 - report procedures
 - ipra-ora-two-report
 - setup
 - support procedures

- ora-two-find-down-nodes-for-root-cause
- ora-two-find-root-cause-for-manager
- ora-two-find-root-cause-for-object
- ora-two-recursive-collect-non-terminal-nodes
- ora-two-recursive-collect-passive-nodes
- ora-two-recursive-collect-terminal-nodes
- ORA-TWO event API
 - domain methods
 - ora-two-recursive-collect-non-terminal-nodes
- ORA-TWO event methods
 - ora-two-fail-method
 - ora-two-recover-method
- ora-two.kb* module
- ora-two-collect-non-terminal-nodes
- ora-two-collect-passive--nodes
- ora-two-collect-terminal-nodes
- ora-two-domain-consistency-check
- ora-two-find-down-nodes-for-root-cause
- ora-two-find-root-cause-for-manager
- ora-two-find-root-cause-for-object
- ora-two-get-state
- ora-two-manager-object
- ora-two-node-type
- ora-two-poll-fail-message
- ora-two-poll-node
- ora-two-poll-recover-method
- ora-two-predicted-poll-fail-message
- ora-two-recursive-collect-non-terminal-nodes
- ora-two-recursive-collect-passive-nodes
- ora-two-recursive-collect-terminal-nodes
- ora-two-root-cause-message
- OV Map Importer
 - about the product
 - adding domain objects
 - attributes of objects
 - batch mode
 - building the domain
 - class definitions used in
 - creating a completion procedure
 - creating a new trap class
 - creating new classes
 - file transfer routines
 - incremental mode
 - initialization objects for IPRA
 - initializations
 - dxiov-mib-lookup
 - XYZ-dxiov-import
 - dxiov-file-retrieve-command
 - dxiov-file-retrieve-name
 - installation
 - merging into applications
 - modes of operation
 - module dependencies
 - network account setup
 - operation of
 - ovobjprint* function
 - testing
 - translation objects
 - file example
 - overloading remote procedure calls
 - overview of trap handling
 - ovobjprint command
 - oxs_sim-request-handler procedure
 - oxs_sim-snmp-event-gsi-obj-symbol
 - initialization
 - oxs_sim-snmp-get-symbol initialization
 - oxs_sim-snmp-get-table-column-symbol
 - initialization
 - oxs_sim-snmp-set-symbol initialization
 - oxs_sim-snmp-sync-gsi-obj-symbol
 - initialization
 - OXS.KB
 - configuring for OV Map Importer
 - oxs-default-unrecognized-trap-class
 - initialization
 - oxs-heartbeat-trap-procedure procedure
 - oxs-running-standalone-toggle initialization
 - oxs-sim-simulate-trap procedure
 - oxs-unrecognized-traps-location
 - oxs-unrecognized-traps-location initialization
- P**
 - PAC-Send-SMH-Message
 - palettes
 - Debugging
 - Decision Blocks
 - External Interfaces
 - General Actions
 - Generic Blocks
 - Local Parameters
 - Opac State Transition
 - Subtask Arguments
 - parameters
 - global
 - asn1_integer

- asn1_octet_string
- snmp_get_req_msg
- snmp_get_rsp_msg
- snmp_getnext_req_msg
- snmp_set_req_msg
- snmp_trap_req_msg
- performance
 - defined
 - mib-maximum-complete-receive-queue-length
 - mib-number-of-completed-receive-discards
 - mib-number-of-completed-receives
 - mib-number-of-expected-val-vs-receive-mismatches
 - mib-number-of-receive-errors
 - mib-number-of-reception-queue-discards
 - snmp-ignored-traps-per-second-min
 - snmp-number-of-ignored-trap-params
 - snmp-number-of-ignored-traps
 - snmp-number-of-request-alarms
 - snmp-number-of-traps-created
 - snmp-traps-per-second
 - snmp-traps-per-second-min
- parser
 - trapd.conf*, *ppd* file
 - trapd_pp* pre-processor utility
- parsing
- passing arguments
- Passport Filter
- passport stamp
- path for filtering log file
- pattern choices
- Pause Capability block
- performance monitoring
- PID
- PID (process ID)
- PID Source
- Ping Manager
 - adding a device
 - configuration file
 - loading
 - sample
 - get configuration status procedure
 - example
 - get-device-status action button
 - installing
 - overview
 - periodic polling
 - starting
- remote procedure calls
 - PM-ADD-DEVICE-CONFIG-RPC
 - PM-CHANGE-IP-ADDRESS-RPC
 - PM-CHANGE-MAX-RETRIES-RPC
 - PM-CHANGE-POLLING-INFO-RPC
 - PM-CHANGE-POLL-INTERVAL-RPC
 - PM-CHANGE-POLL-TIME-OUT-RPC
 - PM-DELETE-DEVICE-CONFIG-RPC
 - PM-DO-DEMAND-POLL-RPC
 - PM-DO-device-POLL-RPC
 - PM-DUMP-AGENT-CONFIG-RPC()
 - PM-GET-DEVICE-CONFIG-RPC
 - PM-KILL-AGENT-RPC()
 - PM-LOAD-CONFIG-FILE-RPC
 - PM-MANAGE-DEVICE-RPC
 - PM-SEND-PING-REQUEST-RPC
 - PM-UNMANAGE-DEVICE-RPC
 - PM-WRITE-CONFIG-FILE-RPC
- running
 - two components of
- pkg_tar.z* file
- Pop General Stack and Delete block
- Pop General Stack Block
- pop general stack of token
- pop the general stack
- pop the stack
- port-number
 - G2-SNMP Bridges
- ppd,preprocessed file names
- preprocessed file (ppd) names
- procedure
- procedure assignments
- Procedure block
- Procedure Statement block
 - referencing the stack
- procedure statements
- Procedure template
- procedures
 - asn1-rfc1212-mib-parser
 - debugging
 - devu-consume-next-field
 - devu-decode-comma-line
 - devu-domain-object-lookup
 - devu-error-handler
 - devu-get-field-n
 - devu-insert-item-in-sorted-ascending-list
 - devu-insert-item-in-sorted-descending-list
 - devu-insert-msg-in-alphabetical-list
 - devu-insert-msg-in-alphabetical-list-allowing-duplicates
 - devu-pattern-matcher-with-indices

devu-safe-text-for-symbol
 devu-safe-window
 devu-strip-linefeeds
 devu-substitute-text
 devu-txt-is-ascii-punctuation-p
 devu-txt-is-ascii-symbol-p
 devu-txt-is-digit-p
 devu-txt-is-mib-character-symbol-p
 mib-clean-up-ws
 mib-create-name-to-oid-translation
 mib-delete-dictionary-on-workspace
 mib-delete-old-list-entries
 mib-enterprise-oid-text-to-name
 mib-read-clears-file
 mib-receiver-create-and-queue
 mib-return-default-msg-category
 mib-substitute-in-format-spec
 mib-translate-info-to-completion-name
 mib-translate-info-to-trap-name
 mib-trapd-preprocessed-conf-reader
 mib-write-clears-file
 opac-get-float-from-stack
 opac-get-integer-from-stack
 opac-get-item-via-text
 opac-get-local-float-var
 opac-get-local-integer-var
 opac-get-local-parameters-as-string
 opac-get-local-text-var
 opac-get-symbol-from-text
 opac-get-text-from-stack
 opac-if-token-error-free
 opac-pop-general-stack
 opac-programmed-start
 opac-replace-local-parms-in-text
 opac-set-local-float-var
 opac-set-local-integer-var
 opac-set-local-text-var
 opac-show-stack-top-from-window
 opac-show-token-info-from-window
 opac-start-task
 opac-task-kill-new-duplicate
 opac-task-kill-old-duplicate
 opac-token-delete
 opac-token-error-handler
 oxs_sim-request-handler
 oxs-heartbeat-trap-procedure
 oxs-sim-simulate-trap
 sample-process-all-clears-for-entries
 sample-send-ov-trap
 snmp-get-mibrec-field-by-name
 snmp-get-mibrec-field-by-oid

snmp-get-mibrec-field-by-pos
 snmp-get-rpc-call
 snmp-non-blocking-set-rpc-call
 snmp-set-rpc-call
 process ID (PID)
 processes

- SNMP Gateway Bridge (SGB)

 Publish Event
 Publish New Event
 publish new events
 Publish Subscribe Mechanism
 Put Connected Objects on Stack block
 Put Float on Stack block
 Put Integer on Stack block
 Put Item on Stack block
 Put Text on Stack block

Q

query event history
 Query Filter
 queues

- clearing
- viewing mib-receiver object

R

reachability analysis

- defined

 read a file
 Read Domain Map
 Read Domain Map block
 Read File block
 receiver procedures

- defined
- G2SNMP_RECEIVE_EOT
- G2SNMP_RECEIVE_FLOAT
- G2SNMP_RECEIVE_INTEGER
- G2SNMP_RECEIVE_MESSAGE
- G2SNMP_RECEIVE_STRING
- G2SNMP_RECEIVE_TRAP_PACKET

 Relationship
 remote procedure calls

- base
 - G2SNMP_BLOCKING_TRANSACTION
 - G2SNMP_NONBLOCKING_TRANSACTION
- blocking
- configuring communication parameters

- filtering traps in SGB
- G2 to GSI
- G2SNMP_ADD_FILTERED_TRAP
- G2SNMP_DELETE_FILTERED_TRAP
- G2SNMP_MODIFY_COMM_PARAMS
- G2SNMP_USE_SNMP_COMM_PARAMS
- G2SNMP_USE_SNMP_DEFAULTS
- GSI to G2
- non-blocking
- overloaded
 - GET_2_BLOCKING
 - GET_BLOCKING_SINGLE
 - GET_NONBLOCKING_SINGLE
 - SEND_NOVAR_TRAP_
 - NONBLOCKING
 - SEND_TRAP_NONBLOCKING
 - SEND_TRAP_STATUS_
 - NONBLOCKING
 - SET_BLOCKING
- overloading
- retrieve the state
- root user
 - becoming
- Run Domain Object Method block
- running a specified method

S

- sample-process-all-clears-for-entries
 - procedure
- sample-send-ov-trap procedure
- script
 - install
- send a message
- Send CDG Event block
- Send SMH Message block
- SEND_NOVAR_TRAP_NONBLOCKIN
 - overloaded remote procedure call
- SEND_NOVAR_TRAP_NONBLOCKING
 - overloaded remote procedure call
- SEND_TRAP_NONBLOCKING overloaded
 - remote procedure call
- SEND_TRAP_STATUS_NONBLOCKING
 - overloaded remote procedure call
- Set Event Attribute
- Set Local Float from Source block
- Set Local Integer from Source block
- Set Local Item from Source block
- Set Local Parameter from Source block
- Set Local Text from Source block

- Set Message Attribute block
- SET_BLOCKING overloaded remote
 - procedure call
- setup
 - configuration
 - outline of steps
- SGB (SNMP Gateway Bridge)
- Show Stack Top block
- Show Token block
- Show Token Info Procedure
- Show Workspace block
- Show Workspace Not Stacked block
- simulation facilities
 - defined
 - SNMP agent MIB
 - configuration
 - configuring tables of
 - configuring variables of
 - SNMP traps
 - configuration
 - configuring variable fields
 - executing
- smh-histories
- SNMP agent MIB simulation facilities
- SNMP Gateway Bridge (SGB)
 - executing
 - port-number
 - ports for G2-SNMP Bridges
 - processes
 - running as a background process
 - SNMP transactions
 - status messages reported by
- SNMP Get block
- SNMP Get request
- SNMP Get Table Column block
- SNMP Set block
- SNMP Set request
- SNMP transactions
 - blocking and non-blocking
 - overloading remote procedure calls
 - remote procedure calls
- SNMP traps
 - simulation of
- snmp_get_req_msg global parameter
- snmp_get_rsp_msg global parameter
- snmp_getnext_req_msg global parameter
- snmp_set_req_msg global parameter
- snmp_trap_req_msg global parameter
- snmp-desc-value function
- snmp-get-mibrec-field-by-name procedure
- snmp-get-mibrec-field-by-oid procedure

- snmp-get-mibrec-field-by-pos procedure
- snmp-get-rpc-call procedure
- snmp-ignored-traps-per-second-min
 - performance parameter
- snmp-non-blocking-set-rpc-call procedure
- snmp-number-of-ignored-trap-params
 - performance parameter
- snmp-number-of-ignored-traps performance parameter
- snmp-number-of-request-alarms performance parameter
- snmp-number-of-traps-created performance parameter
- snmp-set-rpc-call procedure
- snmp-severity-to-status-text-conversion
 - function
- snmp-simulated-trap-receiver object
- snmp-traps-per-second performance parameter
- snmp-traps-per-second-min performance parameter
- spacing
- spawn
- spawn a process
- Spawn No Return block
- Spawn Return Output block
- Spawn Return PID block
- stack
- Stack Operations palette
 - Generic Put Something on Stack
 - Pop General Stack
 - Pop General Stack and Delete
 - Put Connected Objects on Stack
 - Put Float on Stack
 - Put Integer on Stack
 - Put Item on Stack
 - Put Text on Stack
- Start an OPAC procedure
- State Transition diagram
- State Transition Diagram APIs
- State Transition diagram completion
- State Transition Diagrams
- state transition diagrams
- State Transition model
- status messages reported by G2-SNMP Bridges
 - straps* process
- string
- string receiver attributes
 - Ompe-additional-text
 - Ompe-category
 - Ompe-message-text

- subordinate-to
- subroutine
- Subscribers
- Subtask Arguments palette
 - Item Argument
 - Value Argument
- Subtask block
- Subtask Completion block
- Subtask Start block
- superior-to
- SymCure events

T

- target
- Target Attribute Filter
- Target Class Filter
- Task Kill block
- Task Spawn block
- Text
- text
- text parameters
- three-way branching capability
- Time Filter
- token
- token information
- token stack
- token stack top
- top-level object
- Transition Event block
- Transition Event blocks
- trap classes
 - crating from *trapd.conf.ppd* file
- trap handling
 - clears for
 - defining the attribute
 - example procedure
 - manually entering the attribute
 - completion procedure
 - determining
 - example procedure
 - determining trap class to create
 - overview
 - trap class creation
 - unrecognized SNMP traps
 - viewing unrecognized SNMP Traps
 - workspace
- trapd* daemon
- trapd_pp* pre-processor utility
- trapd.conf

- trapd.conf* file
 - severity information
 - text message format
- trapd.conf* trap filter definition file
- trapd.conf.ppd* file
 - parser
- traps
 - adding and deleting filtered traps
 - filtering
 - message structure
 - sending an HP OpenView status
 - sending to external systems
 - snmp-simulated-trap-receiver object
 - trapd.conf* filter definition file
 - turning filtering on and off
- two-way branching capability

- Windows platform
- workspace
- workspace connectivity
- Workspace Spec
- Write File block
- writing the map

U

- unix installation
 - becoming root user
 - determining device name
 - G2-SNMP Bridges from CD-ROM
 - G2-SNMP Bridges from tape
 - install script
 - mounting a CD-ROM
 - unmounting a CD-ROM
- Unix library function
- UNIX operating system
- unmount
 - CD-ROM
- unrecognized SNMP traps handling
- user-defined G2 procedure
- user-defined relation
- user-defined-error-handler initialization
- Using Indirect References

V

- Value Argument block
- variable
- vendor MIBS
- viewing
 - mib-receiver object queues
 - unrecognized SNMP Traps workspace

W

- Wait State Action Procedure
- Wait State block

