G2 Diagnostic Assistant

API Reference

Version 5.1 Rev. 0

G2 Diagnostic Assistant API Reference, Version 5.1 Rev. 0
May 2007

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

Part Number: DOC116-510

Contents

Preface

Describes this manual and the conventions that it uses.

About this Manual

This manual describes the procedures that make up the application programmer's interface (API) for the Gensym Diagram Language (GDL). All programmatic interactions with GDL take place through this set of specially designated "public interface" procedures. You can use the API to modify and control diagrams because these procedures give you programmatic access to otherwise manual functions.

The API consists of G2 procedures and methods. You can access the API by writing your own G2 procedures that call these procedures. In some cases, you may create subclasses of GDL classes (for example, using the custom block wizard) and write methods on your subclasses. For more information about G2 procedures, subclassing, and methods, see the *G2 Reference Manual*.

Note Except for custom block classes, GDL and GDA classes must not be subclassed.

Audience

This document is written for GDA application developers.

It is assumed that the reader is knowledgeable about G2, including how to write procedures and methods; and about GDA, including its class structure.

A Note About the API

The GDA 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 GDA release that includes them.

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

If GDA does not seem to provide the capabilities that you need, contact Gensym Customer Support at 1-781-265-7301 (Americas) or +31-71-5682622 (EMEA) for further information.

Conventions

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

Typographic

Convention Examples	Description
g2-window, g2-window-1, ws-top-level, sys-mod	User-defined and system-defined G2 class names, instance names, workspace names, and module names
history-keeping-spec, temperature	User-defined and system-defined G2 attribute names
true, 1.234, ok, “Burlington, MA”	G2 attribute values and values specified or viewed through dialogs

Convention Examples	Description
Main Menu > Start KB Workspace > New Object create subworkspace Start Procedure	G2 menu choices and button labels
conclude that the x of y ...	Text of G2 procedures, methods, functions, formulas, and expressions
new-argument	User-specified values in syntax descriptions
text-string	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
`c:\Program Files\Gensym\`	Windows pathnames
`/usr/gensym/g2/kbs`	UNIX pathnames
`spreadsh.kb`	File names
`g2 -kb top.kb`	Operating system commands
`public void main()` `gsi_start`	Java, C and all other external code

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

Procedure Signatures

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

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

Related Documentation

G2 Core Technology

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

G2 Utilities

- *G2 ProTools User's Guide*
- *G2 Foundation Resources User's Guide*
- *G2 Menu System User's Guide*
- *G2 XL Spreadsheet User's Guide*
- *G2 Dynamic Displays User's Guide*
- *G2 Developer's Interface User's Guide*
- *G2 OnLine Documentation Developer's Guide*
- *G2 OnLine Documentation User's Guide*

- *G2 GUIDE User's Guide*
- *G2 GUIDE/UIL 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`

1

Block Diagram API

Describes procedures used to manipulate blocks and the block diagram.

Configuring Blocks Programmatically

The GDA user interface provides configuration dialogs for configuring blocks. These configuration dialogs not only conclude the values into the appropriate attributes, but also, in some cases, conduct validation on the values or initialize the blocks.

GDA provides an API that provides access to these same mechanisms that enable the configuration dialogs to properly configure blocks. This is done by calling the gdl-configure procedure, and passing in a sequence that lists the attributes to be changed. These changes occur synchronously, so it is not necessary to include an allow other processing statement to accompany the calls to gdl-configure.

The list of valid attributes can be displayed by attaching the action link of a set attribute block to an instance of the specified blocks, and configuring the set attribute block. In addition, the gdl-get-configuration procedure returns this list of valid attributes, along with their current values.

Note In release notes for previous versions, we provided instructions on how to work around the lack of an API for configuring blocks. That workaround is no longer valid. It is now necessary that you use the API, as specified in this chapter. Failure to do so (that is, concluding the attribute values directly) can result in improperly configured blocks.

Unlike the dialogs in GDA, the gdl-configure procedure does not attempt to validate its inputs before configuring the block. It is the responsibility of the caller to be sure that the attribute values passed to gdl-configure result in a properly configured result. If the block is misconfigured as a result of a call to gdl-configure, the block will not work properly. Some blocks will produce incorrect results, others will error when evaluated, and others will attempt to correct the misconfiguration.

Procedures

This chapter describes the procedures used to perform block and menu actions.

Many of the main block and path menu choices, and main menu choices, are accessible through the API. Some of the procedures give you access to the inner-workings of GDA, exposing functionality not normally visible to the user. These procedures concern the compiling of diagrams.

The procedures can be categorized as shown in the following table:

Category	Procedures
Block and path menu choices	gdl-clear-block-error gdl-disable-evaluation gdl-enable-evaluation gdl-evaluate-block gdl-generate-explanation gdl-get-sensor-of-entry-point gdl-reset-item gdl-update-g2-chart
Creating diagrams	gdl-attach-capability gld-attach-clock gdl-attach-restriction gdl-clone-item gdl-create-item gdl-delete-item gdl-make-connection-block-to-block gdl-make-connection-block-to-stub gdl-make-connection-stub-to-block gdl-make-connection-stub-to-stub gdl-transfer-item
Running diagrams	gdl-block-evaluation-is-enabled gdl-compile-block-diagram
Global functions	gdl-disable-data-input gdl-enable-data-input gdl-force-total-recompile gdl-reset-all-gdl-objects gdl-turn-animation-off gdl-turn-animation-on

For integration with GDL paths, see Chapter 2, “Custom Block Evaluator API,”on the use of the get, set, and propagate procedures. These are intended primarily for use in custom block evaluators, but can also be used independently.

Most procedures require a ui-client-item as their last argument. In general, this is the g2-window or telewindows 2 window where the call originated. Passing the client identifies for whom the action is being carried out. You can access user mode, language, and other properties of the client through this argument.

When you begin a thread of processing from an action button or user menu choice, you can access the window by using the "this window" syntax in the action of the button or menu choice.

When you start a procedure through a rule, you must associate a window with a thread of processing by using the window gfr-default-window as the window argument.

gdl-attach-capability

Attaches a capability to a block.

Synopsis

gdl-attach-capability
(*Cap*: class gdl-capability, *Blk*: class gdl-block, *Sym2*: symbol,
Pos2: integer, *Vertices*: class integer-list, *InvalidateWS*: truth-value,
Client: class object)
–> *Connection*: class gdl-capability-link

Arguments	Description
Cap	The capability to be attached.
Blk	The block to which the capability is attached at a newly created connection.
Sym2	A symbol specifying a side on *Blk*.
Pos2	The position of the new connection on *Blk*.
Vertices	An integer list of vertices that specify the bends of the connection as described in the *G2 Reference Manual*.
InvalidateWS	A value of true instructs the procedure to invalidate the workspaces on which the connection was made.
Client	The g2-window or client object originating this call.

Return Value	Description
Connection	The newly created connection instance.

Description

This procedure creates a connection from an existing stub on *Cap*, creating a new connection on *Blk*. This procedure only works properly with capabilities. The connection returned is transient.

The connection must be made from an existing stub to a gdl-block. The stub on *Cap* is not specified by the parameters of gdl-attach-capability because the call assumes the capability is standard; it has a single, default, capability link stub on

the bottom side of its icon. If there are multiple capability link stubs on *Cap*, GDL signals an error. The gdl-attach-capability call makes the connection only if the stub on *Cap* is not already connected. If the connection is occupied, GDL signals an error.

When gdl-attach-capability makes a connection, the new connection is returned. The connection is located on *Blk* on the side specified by *Sym2* at the position specified by *Pos2*. Attaching capabilities ignore any existing capability link stubs on *Blk*.

By passing a list of vertices to gdl-attach-capability, you can specify vertices on the new connection. To specify no vertices or to use the default vertices that G2 creates, pass an empty list as the *Vertices* argument.

Populate the list *Vertices* with integers that indicate the length of the connection before a bend occurs. To indicate the direction of the bend, use positive or negative integers. For more information on lists, see the *G2 Reference Manual*.

Caution This procedure does not delete the list of vertices passed to it. Failure to delete the list after the procedure returns results in a leaked item.

The *InvalidateWorkspace* flag schedules the workspace of the newly created connection for recompilation, as described in gdl-compile-block-diagram. Proper initialization of gdl-connections occurs during the compilation process. For a newly created connection to pass or display data, set the *InvalidateWorkspace* flag to true or make a separate call to gdl-compile-block-diagram.

gdl-attach-clock

Attaches a clock to a block.

Synopsis

gdl-attach-clock
(*Clock*: class gdl-clock, *Blk*: class gdl-block, *Sym2*: symbol, *Pos2*: integer, *Vertices*: class integer-list, *InvalidateWS*: truth-value, *Client*: class object)
–> *Connection*: class gdl-capability-link

Arguments	Description
Clock	The clock to be attached.
Blk	The block to which the clock is attached at a newly created connection.
Sym2	A symbol specifying a side on *Blk*.
Pos2	The position of the new connection on *Blk*.
Vertices	An integer list of vertices that specify the bends of the connection as described in the *G2 Reference Manual*.
InvalidateWS	A value of true instructs the procedure to invalidate the workspaces on which the connection was made.
Client	The g2-window or client object originating this call.

Return Value	Description
Connection	The newly created connection instance.

Description

This procedures behaves exactly like the gdl-attach-capability procedure except that it attaches a clock to a block. See gdl-attach-capability on page 5.

gdl-attach-restriction

Attaches a restriction to a block.

Synopsis

gdl-attach-restriction
(*Rest*: class gdl-restriction, *Blk*: class gdl-block, *Sym2*: symbol,
Pos2: integer, *Vertices*: class integer-list, *InvalidateWS*: truth-value,
Client: class object)
–> *Connection*: class gdl-restriction-link

Arguments	Description
Rest	The restriction to be attached.
Blk	The block to which the restriction is attached at a newly created connection.
Sym2	A symbol specifying a side on *Blk*.
Pos2	The position of the new connection on *Blk*.
Vertices	An integer list of vertices that specify the bends of the connection as described in the *G2 Reference Manual*.
InvalidateWS	A value of true instructs the procedure to invalidate the workspaces on which the connection was made.
Client	The g2-window or client object originating this call.

Return Value	Description
Connection	The newly created connection instance.

Description

This procedure creates a connection from an existing stub on *Rest*, creating a new connection on *Blk*. This procedure only works properly with restrictions. The connection returned is transient.

The connection must be made from an existing stub to a gdl-block. The stub on *Rest* is not specified by the parameters of gdl-attach-restriction because the call assumes the restriction is standard: it has a single, default, restriction link stub on

the bottom side of its icon. If there are multiple restriction link stubs on *Rest*, GDL signals an error. The gdl-attach-restriction call makes the connection only if the stub on *Rest* is not already connected. If the connection is occupied, GDL signals an error.

When gdl-attach-restriction makes a connection, it returns the new connection. The connection is located on *Blk* on the side specified by *Sym2* at the position specified by *Pos2*. Attaching restrictions ignores any existing restriction link stubs on *Blk*.

By passing a list of vertices to gdl-attach-restriction, you can specify vertices on the new connection. To specify no vertices or to use the default vertices that G2 creates, pass an empty list as the *Vertices* argument.

Populate the *Vertices* list with integers that indicate the length of the connection before a bend occurs. To indicate the direction of the bend, use positive or negative integers. For more information, see the *G2 Reference Manual*.

Caution This procedure does not delete the list of vertices passed to it. Failure to delete the list after the procedure returns results in a leaked item.

The *InvalidateWorkspace* flag schedules the workspace of the newly created connection for recompilation, as described in gdl-compile-block-diagram. Proper initialization of gdl-connections occurs during the compilation process. For a newly created connection to pass or display data, set the *InvalidateWorkspace* flag to true or make a separate call to gdl-compile-block-diagram.

gdl-block-evaluation-is-enabled

A function that can be used to determine whether or not a block has evaluation enabled.

Synopsis

gdl-block-evaluation-is-enabled
(*Blk*: class gdl-block)
–> *Value*: truth-value)

Argument	Description
Blk	The block in question.

Return Value	Description
Value	A truth-value indicating whether the block has evaluation enabled.

Description

The call is used to determine if a block has had evaluation disabled. It returns true if the block is enabled and false if the block is disabled.

gdl-clear-block-error

Clears the error state of a block.

Synopsis

gdl-clear-block-error
(*Obj*: class gdl-object, *Client*: class object)

Arguments	Description
Obj	The object in error to be cleared.
Client	The g2-window or client object originating the error.

Description

This procedure is the equivalent of the clear error menu choice on a block.

gdl-clone-item

Creates a clone of an item.

Synopsis

gdl-clone-item
(*Item*: class item, *Client*: class object)
–> *Item*: class item

Arguments	Description
Item	The item to be cloned.
Client	The g2-window or client object originating this call.

Return Value	Description
Item	The newly cloned instance.

Description

This procedure creates an instance by cloning the passed item. The newly cloned item is transient.

gdl-compile-block-diagram

Schedules a diagram for compilation.

Synopsis

gdl-compile-block-diagram
(*Workspace*: class kb-workspace, *Client*: class object)

Argument	Description
Workspace	The KB workspace containing the diagram to be compiled
Client	The g2-window or client object originating this call

Description

This procedure places the workspace on a queue to be compiled. GDL schedules the recompilation after the currently running tasks have completed. The process of queuing a workspace for recompilation is called "invalidating" the workspace.

Use this procedure on a workspace after the changes you made using the API are complete. Calling this procedure enables GDL to account for the new configuration in its next computation cycle. Failing to call this procedure can cause errors, particularly if you have deleted items.

If you make programmatic changes to a diagram with procedures that have an Invalidate Workspace argument set to true, you do not need to call the gdl-compile-block-diagram procedure. If you are only making a single change to a diagram, you can invalidate the workspace during the call that makes the change.

If, using this procedure, you make many changes to a diagram contained on a single workspace, set Invalidate Workspace to false and call gdl-compile-block-diagram after the changes are complete. Although no errors result from multiple invalidations of the same diagram, multiple invalidations could significantly slow the execution of your diagrams.

Caution This procedure schedules a workspace for recompilation but does not launch a synchronous recompile of the specified workspace. The recompile is asynchronous and may not complete before the execution of subsequent calls in the procedure. Thus, a call to gdl-compile-block-diagram does not mean that the new configuration is available for subsequent calls in the procedure.

See the "Examples" on page 32 for "gdl-make-connection-block-to-block."

gdl-configure

Synopsis

gdl-configure
(*Item*: item, *Attributes*: sequence, *Client*: ui-client-item)

Arguments	Description
Item	The GDA item to be configured.
Attributes	A sequence of structures, one for each attribute, containing the name of the attribute and the new value.
Client	The g2-window or client object originating this call.

Description

This procedure configures *Item*, which, typically, is a subclass of gdl-object. The gdl-configure procedure is also used for configurable GDA items that are not objects (e.g., Alarm Readout). This procedure is the programmatic equivalent of using a configuration dialog.

When calling this procedure, you must create a sequence of structures. Each structure contains two attributes, which are used to specify an attribute on *Item* that is to be changed. The following table specifies these attributes:

Attribute	Type	Description
attribute-name	symbol	Contains the name of the attribute that exists on the item. The attribute should be named exactly as it is named on the item itself.
attribute-value	value	The value of the attribute named by *attribute-name*.

It is the responsibility of the calling procedure to ensure that the attribute named in the structure exists for the class of *Item*, and that the data type of the data in *attribute-value* is the correct data type for the attribute on *Item*. If these are not properly checked, an error is signaled when gdl-configure attempts to conclude the values.

To obtain the list of valid attributes for a particular class, call gdl-get-configuration, which returns the sequence of all valid attributes in the same format as that required for this procedure.

gdl-configure can also be called with *Attributes* passed as an empty sequence. This does not mean that no items should be reconfigured; instead, it indicates that any or all of the attributes on *Item* may have been reconfigured, and so any other configuration (besides the concluding of values into the appropriate attribute) which must take place should do so, on the basis that the attributes have changed.

See Also

"gdl-get-configuration" on page 27

"gdl-object::gdl-configure" on page 111

gdl-create-item

Creates an instance of a given class.

Synopsis

gdl-create-item
(*ClassName*: symbol, *Client*: class object)
–> *Item*: class item

Argument	Description
ClassName	The name of the class to be instantiated.
Client	The g2-window or client object originating this call.

Return Value	Description
Item	The newly created instance

Description

This procedure creates an instance of the class named by *ClassName*. The instance is created from a palette, if one exists. Otherwise, an instance of the named class is created. The newly created instance is properly initialized. It is not made permanent.

Standard GDA blocks are created from the palette. If you use gdl-create-item to create an instance of a user-defined custom block, where you have placed that block on a palette using the custom class wizard, the new block contains proper initialization and subworkspaces.

The gdl-create-item procedure uses gfr-palette behavior to control the creation of instances. For more information on palettes, palette management, and how to customize behavior, see the *G2 Foundation Resources User's Guide*.

Caution This procedure works with GDA related items only. Using gdl-create-item to create an item from another module or hierarchy (for example, GUIDE) might produce an uninitialized instance.

Example

The following sample code creates an instance of the GDA observation class High and Low:

```
NewBlock = call gdl-create-item(the symbol gdl-hi-lo-limit-block,
    gfr-default-window);
```

The item NewBlock is transient and does not exist on any workspace. To transfer the new block to a workspace named by the local variable Wksp at the location 75, 75 and make it a permanent part of your diagram, use the following code:

```
NewBlock = call gdl-create-item (the symbol gdl-hi-lo-limit-block,
    gfr-default-window);
call gdl-transfer-item(NewBlock, Wksp, 75,75, FALSE, gfr-default-window);
make NewBlock permanent;
call gdl-compile-block-diagram (Wksp, gfr-default-window);
```

The result is a permanent, initialized, and compiled block of the class High and Low deposited on the indicated workspace. This instance contains the proper encapsulation subworkspace for a block of this class, as if you manually cloned the item from the palette.

For another example of the use of gdl-create-item, see "gdl-make-connection-block-to-block" on page 30.

gdl-delete-item

Deletes an item.

Synopsis

gdl-delete-item
(*Item*: class item, *IgnorePermanence*: truth-value,
InvalidateWorkspace: truth-value, *Client*: class object)

Arguments	Description
Item	The item to be deleted
IgnorePermanence	A value of true instructs the procedure to delete items whether or not they are permanent.
InvalidateWorkspace	A value of true instructs the procedure to put the workspace on the queue for recompilation after deleting the item.
Client	The g2-window or client object originating this call.

Description

This procedure deletes the item passed to it. When called, this procedure properly handles related items within GDA. For example, when deleting an encapsulation block, gdl-delete-item properly handles the encapsulation hierarchy.

When you are deleting a block connected to other blocks, gdl-delete-item deletes the selected block and its connections to other blocks without changing the permanence of the connected blocks.

If you pass false as the value of the *IgnorePermanence* argument, the resulting behavior is similar to the delete action in G2. The programmatic delete in G2 only works on transient items. Consequently, if you pass false as the value of the *IgnorePermanence* argument for gdl-delete-item, GDL signals an error when asked to delete a permanent item. If you pass true as the value of the *IgnorePermanence* argument, gdl-delete-item deletes the item in question regardless of permanence.

Using the *InvalidateWorkspace* flag schedules the workspace of the item for recompilation, as described in gdl-compile-block-diagram.

gdl-disable-data-input

Disables data input.

Synopsis

gdl-disable-data-input
(*Client*: class object)

Argument	Description
Client	The g2-window or client object originating this call.

Description

This procedure turns off the Enable Data Input option on the main menu. The gdl-disable-data-input procedure stops:

- Data flow into entry points.
- The execution of signal generators.
- Clock capabilities.

gdl-disable-evaluation

Disables the evaluation of a block.

Synopsis

gdl-disable-evaluation
(*Blk*: class gdl-block, *Client*: class object)

Arguments	Description
Blk	The block to be disabled.
Client	The g2-window or client object originating this call.

Description

This procedure is the equivalent of the disable evaluation menu choice on a block. Disabling a block with attached capabilities also disables the attached capabilities. Disabling an encapsulation block disables all the blocks in its workspace hierarchy.

gdl-enable-data-input

Enables data input.

Synopsis

gdl-enable-data-input
(*Client*: class object)

Argument	Description
Client	The g2-window or client object originating this call.

Description

This procedure turns on the Enable Data Input option on the main menu. The gdl-enable-data input procedure starts:

- Data flow into entry points.
- The execution of signal generators.
- Clock capabilities.

gdl-enable-evaluation

Enables the evaluation of a disabled block.

Synopsis

gdl-enable-evaluation
(*Blk*: class gdl-block, *Client*: class object)

Arguments	Description
Blk	The block to be enabled.
Client	The g2-window or client object originating this call.

Description

This procedure is the equivalent of the enable evaluation menu choice on a block. Enabling a block with attached capabilities also enables the attached capabilities. Enabling an encapsulation block enables all the blocks in its workspace hierarchy.

Attempting to call gdl-enable-evaluation on a block whose superior encapsulation is disabled results in a signalled error.

gdl-evaluate-block

Schedules a block for evaluation.

Synopsis

gdl-evaluate-block
(*Blk*: class gdl-block, *Client*: class object)

Arguments	Description
Blk	The block to be evaluated.
Client	The g2-window or client object originating this call.

Description

This procedure is the equivalent of the evaluate menu choice on a block. This procedure schedules the block for an evaluation during a future cycle of GDL.

Caution This procedure schedules a block for evaluation only. It does not force a synchronous evaluation of the block. A call to gdl-evaluate-block does not mean that the block executes before the subsequent statement of your procedure.

gdl-force-total-recompile

Completely recompiles an application.

Synopsis

gdl-force-total-recompile ()

Description

This procedure is available for the case of corrupted applications. It forces recompilation of all workspaces, objects and paths. Any data stored in gdl-paths or in the states of blocks is lost.

gdl-generate-explanation

Synopsis

gdl-generate-explanation
(*Blk*: gdl-block-with-explainer, *Client*: class object)
–> *text*

Arguments	Description
Blk	The block whose status is to be explained.
Client	The client for this call.

Return Value	Description
text	Text that explains the status of *Blk*.

Description

This procedure returns the explanation associated with the status of a block and upstream blocks. The returned text is the text that would be posted to the explanation queue when the current explanation menu choice is selected.

gdl-get-active-setting

Synopsis

gdl-get-active-setting
(*ClassName*: symbol, *Client*: ui-client-item)
–> *gfr-module-setting*

Argument	Description
ClassName	The class of the module setting to be retrieved.
Client	The client originating this call.

Return Value	Description
gfr-module-setting	A module setting of the type named by *ClassName*.

Description

This procedure returns the active module setting object of a given class, if one exists in the top level module. If more than one instance of the class exists in the top level module, one instance is chosen arbitrarily. If no instance of the class exists in the top level module, this procedure creates one by cloning the setting returned by gfr-get-active-setting. The new instance is deposited in the gfr-public bin and the settings for the top level module are reinstalled.

The class name of the returned item matches the *ClassName* argument exactly (i.e. the returned object is not a subclass of the target class). If there is no instance of the given *ClassName*, the gfr-get-active-setting error is signaled.

gdl-get-configuration

Synopsis

gdl-get-configuration
(*Item*: item, *Client*: ui-client-item)
–> *sequence*

Description

Argument	Description
Item	The item for which a configuration is requested.
Client	The window or client originating this call.

Return Value	Description
sequence	A sequence of structures, one for each configurable attribute, containing the name of the attribute and the new value.

This procedure retrieves the current configuration of the GDA item, *Item*. Typically, *Item* is a subclass of gdl-object, but the gdl-get-configuration procedure can also be used for GDA items that are not objects. Specifically, any item with an associated gdl-dialog-intermediary, as well as any GDA subclass of gfr-module-setting, can be accessed with gdl-get-configuration. This procedure returns a list of all the configurable attributes (configurable using gdl-configure) and their current values. This API provides similar information to the dialog for the Set Attribute block.

The return value for this procedure is a sequence of structures. Each structure contains at least two attributes, sometimes, three, which serve to specify which attributes are configurable and what the current value for that attribute is. The following table specifies the form of the structures in the returned sequence:

Attribute	Type	Description
attribute-name	symbol	Contains the name of the attribute that exists on *Item*. The attribute should be named exactly as it is named on the item itself.
attribute-value	value	The value of the attribute named by *attribute-name*.
elements	sequence	An additional attribute, not present for all attributes or all classes. If the attribute named by *attribute-name* has a list of allowable values, these values can be supplied in the sequence *elements*.

By default, if a particular attribute has no value, the *attribute-value* returned is the symbol NONE.

See Also

"gdl-configure" on page 14

"gdl-object::gdl-configure" on page 111

"gdl-object::gdl-get-configuration" on page 113

gdl-get-sensor-of-entry-point

Returns the sensor of an entry point, if one exists.

Synopsis

gdl-get-sensor-of-entry-point
(*EntryPoint*: class gdl-entry-point, *Client*: class object)
–> *Variable*: item-or-value

Arguments	Description
EntryPoint	The entry point whose sensor is sought.
Client	The g2-window or client object originating this call.

Return Values	Description
Variable	The variable or parameter of the sensor.

Description

This procedure returns the variable or parameter that is the sensor of *EntryPoint*, if one exists. The gdl-get-sensor-of-entry-point procedure is the programmatic equivalent of the go to sensor menu choice on entry points. For more information on the operation of entry points, see the *GDA Reference Manual*.

If no sensor exists for *EntryPoint*, gdl-get-sensor-of-entry-point returns the symbol NONE.

Note A link between an entry point and its sensor is only made when the configuration of that entry point is set to external. Using this procedure to obtain the sensor of an entry point which is configured to use embedded data may return an object other than the one named in the name of sensor attribute.

gdl-make-connection-block-to-block

Creates a directed connection between two objects, creating new connection stubs at specified locations.

Synopsis

gdl-make-connection-block-to-block
(*Obj1*: class object, *Sym1*: symbol, *Pos1*: integer, *Obj2*: class object,
Sym2: symbol, *Pos2*: integer, *ClassName*: symbol,
Vertices: class integer-list, *InvalidateWS*: truth-value, *Client*: class object)
–> *Connection*: class gdl-connection

Arguments	Description
Obj1	The object on which to create a new outward-directed connection. *Obj1* cannot be a gdl-block; an error is signaled if it is.
Sym1	A symbol specifying a side on *Obj1* on which the new connection is to be made.
Pos1	The position of the connection on *Obj1*.
Obj2	The object on which to create a new inward-directed connection.
Sym2	A symbol specifying a side on *Obj2* on which the new connection is to be made.
Pos2	The position of the connection on *Obj2*.
ClassName	A symbol that contains the name of the class of connection to be created.
Vertices	An integer list of vertices that specify the bends of the connection.
InvalidateWS	A value of true invalidates the workspace on which the connection is made and directs recompilation of the diagram.
Client	The g2-window or client object originating this call.

Return Value	Description
Connection	The newly created connection instance.

Description

This procedure creates a connection between two objects, *Obj1* and *Obj2*. The connection has direction output from *Obj1* to *Obj2*. The newly created connection is returned and is transient.

The connection must be created from an object that allows the addition of new output connections (subclasses of gdl-block do not allow the addition of output paths).

Connections are uniquely specified by a symbol and a position. The symbol must be a:

- **side** - the top, bottom, left, or right side on which the connection exists.

The position, an integer, can be used to further specify where the connection is located. For example, if the symbol is a side and there are multiple connections on that side, then the connection located at the specified position on that side is used.

The connection on *Obj1* is created on the side given by *Sym1* at the location on that side specified by the integer *Pos1*.

If *Obj2* is not a valid recipient of a gdl-connection (if *Obj2* is a subclass of gdl-block and not a peer block), GDL signals an error and a new connection cannot be made.

If the inputs to gdl-make-connection-block-to-block specify valid connections, the procedure makes a connection of the class specified by *ClassName*. If the symbol *ClassName* does not denote a gdl-connection, GDL signals an error. When GDL makes a connection, it returns the new connection.

By passing a list of vertices to gdl-make-connection-block-to-block, you can specify vertices on the new connection. To specify no vertices or to use the default vertices that G2 creates, pass an empty list as the *Vertices* argument.

Populate the *Vertices* list with integers that indicate the length of the connection before a bend occurs. To indicate the direction of the bend, use positive or negative integers. For more information about creating a connection with vertices, see the *G2 Reference Manual*.

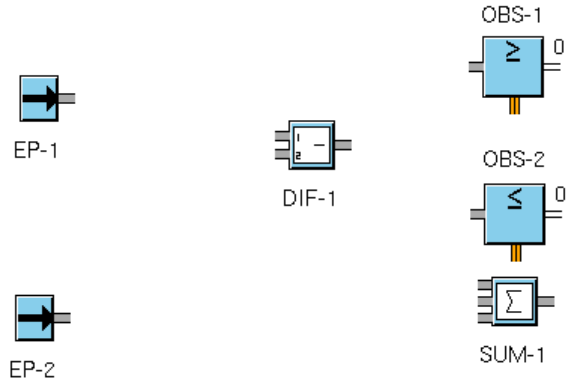
Caution This procedure does not delete the list of vertices passed to it. Failure to delete the list after the procedure returns results in a leaked item.

The *InvalidateWS* flag indicates whether the workspace of the newly created connection is to be scheduled for recompilation, as described in gdl-compile-block-diagram. Proper initialization of gdl-connections occurs during the compilation process. For a newly created connection to pass or display data, set the *InvalidateWS* flag to true or make a separate call to gdl-compile-block-diagram.

Examples

These examples connect blocks and stubs using the four connection procedures. The examples use the following diagram, which shows these blocks:

- Two Numeric Entry Points
- Difference
- High Value Observation
- Low Value Observation
- Summation



Example 1

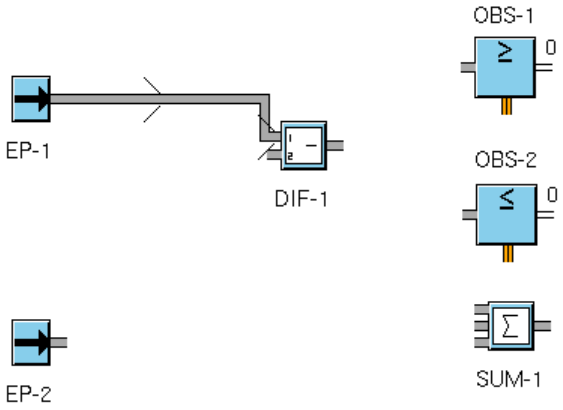
The makeconnstub2stub example procedure connects a Numeric Entry Point to the Difference block using portnames. The gdl-make-connection-stub-to-stub API procedure connects the dp-out port on the Numeric Entry Point to the dp-in-1 port on the Difference block.

This procedure makes these connections permanent so they are not deleted in a reset.

```
makeconnstub2stub (Wksp: class kb-workspace, Win: class g2-window)
VList: class integer-list;
Conn: class gdl-data-path;

begin
    create an integer-list VList;
    Conn = call gdl-make-connection-stub-to-stub
        (ep-3, the symbol dp-out, -1, dif-1, the symbol dp-in-1,
        -1, VList, false, Win);
    make Conn permanent;
    delete VList;
    call gdl-compile-block-diagram (Wksp, Win);
end
```

Executing the makeconnstub2stub example procedure makes the connections shown in the following figure:



Example 2

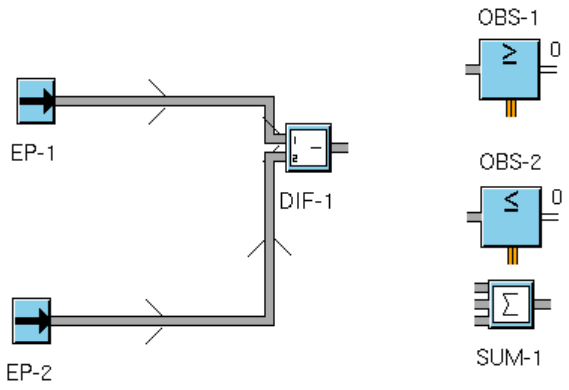
The makeconnstub2stubalt example procedure connects the second Numeric Entry Point (EP-2) to the Difference block, specifying the stubs by side rather than by position, as the makeconnstub2stub example procedure does. The gdl-make-connection-stub-to-stub API procedure connects the specified position on the right side of the Numeric Entry Point to a position on the left side of the Difference block.

This procedure makes these connections permanent so that they are not deleted in a reset.

```
makeconnstub2stubalt (Wksp: class kb-workspace, Win: class g2-window)
VList: class integer-list;
Conn: class gdl-data-path;

begin
    create an integer-list VList;
    Conn = call gdl-make-connection-stub-to-stub (EP-2, the symbol right,
        15, Dif-1, the symbol left, 22,VList, FALSE, Win);
    make Conn permanent;
    delete VList;
    call gdl-compile-block-diagram (Wksp, Win);
end
```

Executing the makeconnstub2stubalt example procedure makes the connections shown in the following figure:



Example 3

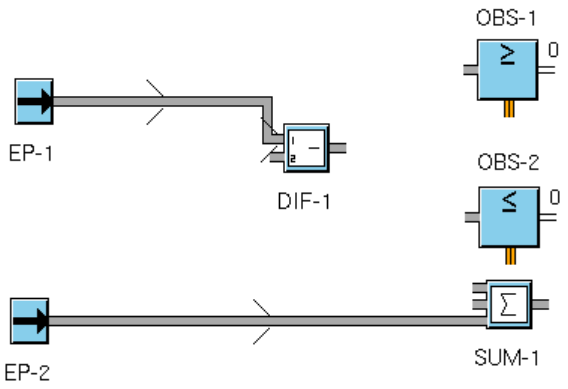
If you want to make a connection to a peer input block, where the portnames are not unique, you must specify the location of the port. The makeconn2summation example procedure connects the second Numeric Entry Point (EP-2) to the Summation block using a portname and a side because the Summation block does not offer a unique reference for a portname. The gdl-make-connection-stub-to-stub API procedure connects the output stub of the Numeric Entry Point to the bottom input stub of the Summation block.

This procedure makes these connections permanent so that they are not deleted in a reset.

```
makeconn2summation (Wksp: class kb-workspace, Win: class g2-window)
VList: class integer-list;
Conn: class gdl-data-path;

begin
    create an integer-list VList;
    Conn = call gdl-make-connection-stub-to-stub (EP-2, the symbol dp-out,
       -1, sum-1 , the symbol dp-in, 27,VList, FALSE, Win);
    delete VList;
    make Conn permanent;
    call gdl-compile-block-diagram (Wksp, Win);
end
```

Executing the makeconn2summation example procedure makes the connections shown in the following figure (the connection made by makeconnstub2stubalt has been deleted):



Example 4

The following example procedure, makeconnjunct, connects the Difference block to three downstream blocks by creating two Junction blocks and connecting the Difference block to the downstream blocks through the Junction blocks.

Using the gdl-make-connection-stub-to-block API procedure, the makeconnjunct example procedure connects the stub on the Difference block to a Junction block and, by using gdl-make-connection-block-to-stub, connects the Junction block to a stub on the High Observation block (OBS-1).

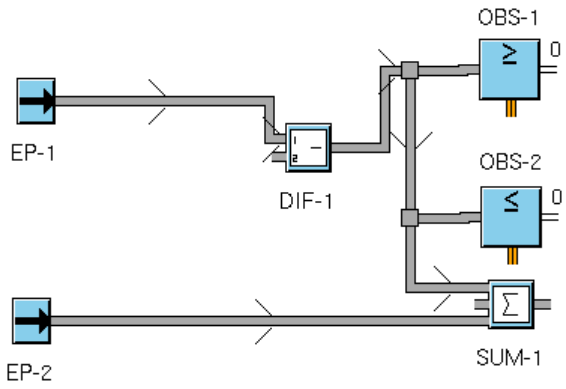
The example procedure connects the first Junction block to a second Junction block by using gdl-make-connection-block-to-block and completes the connection to the Low Observation block (OBS-2) by using gdl-make-connection-block-to-stub.

This procedure makes these connections permanent so that they are not deleted in a reset.

```
makeconnjunct (Wksp:class kb-workspace, Win: class g2-window)
Xpos, Ypos, Xpos1, Xpos2: integer;
JunClass: symbol = symbol (the text of the junction-block of gdl-data-path);
Junct1, Junct2: class default-junction;
StubH: integer;
VList: class integer-list;
Conn: class gdl-data-path;

begin
  Xpos1 = the item-x-position of Dif-1;
  Xpos2 = the item-x-position of Obs-1;
  Ypos = the item-y-position of Obs-1;
  Xpos = Xpos1 + round((Xpos2 - Xpos1)/2);
  Junct1 = call gdl-create-item (JunClass, Win);
  call gdl-transfer-item (Junct1, Wksp, Xpos, Ypos, FALSE, Win);
  make Junct1 permanent;
  StubH = round(the item-height of Junct1/2);
  create an integer-list VList;
  Conn = call gdl-make-connection-stub-to-block (Dif-1, the symbol dp-out, -
     1, Junct1, the symbol left, StubH, VList, FALSE, Win);
  make Conn permanent;
  Conn = call gdl-make-connection-block-to-stub (Junct1, the symbol right,
     StubH, Obs-1, the symbol dp-in, -1, VList, FALSE, Win);
  make Conn permanent;
  Xpos2 = the item-x-position of Obs-1;
  Ypos = the item-y-position of Obs-2;
  Xpos = round(Xpos1 + (Xpos2 - Xpos1)/2);
  Junct2 = call gdl-create-item (JunClass, Win);
  call gdl-transfer-item (Junct2, Wksp, Xpos, Ypos, FALSE, Win);
  make Junct2 permanent;
  Conn = call gdl-make-connection-block-to-block (Junct1, the symbol
     bottom, StubH, Junct2, the symbol top, StubH, the symbol gdl-data-
     path, VList, FALSE, Win);
  make Conn permanent;
  Conn = call gdl-make-connection-block-to-stub (Junct2, the symbol right,
     StubH, Obs-2, the symbol dp-in, -1, Vlist, FALSE, Win);
  make Conn permanent;
  Conn = call gdl-make-connection-block-to-stub (Junct2, the symbol
     bottom, StubH, Sum-1, the symbol dp-in, 5, Vlist, FALSE, Win);
  make Conn permanent;
  delete VList;
  call gdl-compile-block-diagram (Wksp, Win);
end
```

Executing the makeconnjunct example procedure makes the connections shown in the following figure:



gdl-make-connection-block-to-stub

Creates a directed new connection from one object to an existing input stub on a second object.

Synopsis

gdl-make-connection-block-to-stub
(*Obj1*: class object, *Sym1*: symbol, *Pos1*: integer, *Obj2*: class object,
Sym2: symbol, *Pos2*: integer, *Vertices*: class integer-list,
InvalidateWS: truth-value, *Client*: class object)
–> *Connection*: class gdl-connection

Arguments	Description
Obj1	The object on which to create a new, outward-directed connection.
Sym1	A symbol specifying a side on *Obj1*.
Pos1	The position of the new connection on *Obj1*.
Obj2	The object on which to create a connection to an inward-directed stub.
Sym2	A symbol specifying either a portname or a side on *Obj2*.
Pos2	The position of the stub on *Obj2*.
Vertices	An integer list of vertices which specify the bends of the connection as described in the *G2 Reference Manual*.
InvalidateWS	A value of true instructs the procedure to invalidate the workspace on which the connection was made.
Client	The g2-window or client object originating this call.

Return Value	Description
Connection	The newly created connection instance.

Description

This procedure creates a connection between one object, *Obj1*, and an existing stub on a second object, *Obj2*. This procedure only works properly on connections that are used in GDL (subclasses of gdl-connection). The connection returned is transient.

The connection must be created from an object that allows the addition of new output connections to an existing stub (subclasses of gdl-block do not allow the addition of output paths). The connection is created on the side given by *Sym1* at the location on that side specified by the integer *Pos1*.

If the stub on *Obj2* is not uniquely specified (for example, peer-input-blocks often do have multiple inputs with the same portname), then gdl-make-connection-block-to-stub makes a connection to one of the free connection ports.

Stubs are uniquely specified by a *symbol* and a *position*. The symbol can be a:

- **portname** - the name of the port at which the connection stub is attached.
- **side** - the top, bottom, left, or right side on which the stub exists.

The position, an integer, can be used to further specify which stub makes the connection. For example:

- If the symbol is a side and there are multiple connection stubs on that side, then the connection located at the specified position on that side is used.
- If the symbol is a portname and there are multiple stubs with that portname, then the stub located at the specified position, on any side, is used.

If the inputs to gdl-make-connection-block-to-stub specify valid connection stubs, gdl-make-connection-block-to-stub makes a connection only if the connections are not already connected and both connections are of the same class. If either of these conditions are not met, GDL signals an error. When GDL makes a connection, it returns the new connection.

By passing a list of vertices to gdl-make-connection-block-to-stub, you can specify vertices on the new connection. To specify no vertices or to use the default vertices that G2 creates, pass an empty list into the *Vertices* argument.

Populate the *Vertices* list with integers that indicate the length of the connection before a bend occurs. To indicate the direction of the bend, use positive or negative integers. For more information, see the *G2 Reference Manual*.

Caution This procedure does not delete the list of vertices passed to it. Failure to delete the list after the procedure returns results in a leaked item.

The *InvalidateWorkspace* flag schedules the workspace of the newly created connection for recompilation, as described in gdl-compile-block-diagram. Proper initialization of gdl-connections occurs during the compilation process. For a

newly created connection to pass or display data, set the *InvalidateWorkspace* flag to true or make a separate call to gdl-compile-block-diagram.

Example

See "gdl-make-connection-block-to-block" on page 30.

gdl-make-connection-stub-to-block

Creates a directed connection from existing output stub on one object, creating a new connection on the second.

Synopsis

gdl-make-connection-stub-to-block
(*Obj1*: class object, *Sym1*: symbol, *Pos1*: integer, *Obj2*: class object,
Sym2: symbol, *Pos2*: integer, *Vertices*: class integer-list,
InvalidateWS: truth-value, *Client*: class object)
–> *Connection*: class gdl-connection)

Arguments	Description
Obj1	The object on which to create a connection from an outward-directed stub.
Sym1	A symbol specifying either a portname or a side on *Obj1*.
Pos1	The position of the stub on *Obj1*.
Obj2	The object on which to create a new connection at a specified location.
Sym2	A symbol specifying a side on *Obj2*.
Pos2	The position of the new connection on *Obj2*.
Vertices	An integer list of vertices that specify the bends of the connection as described in the *G2 Reference Manual*.
InvalidateWS	A value of true instructs the procedure to invalidate the workspace on which the connection was made.
Client	The g2-window or client object originating this call.

Return Value	Description
Connection	The newly created connection instance.

Description

This procedure creates a connection from an existing stub on *Obj1*, creating a new connection on *Obj2*. This procedure only works properly on connections that are used in GDL (subclasses of gdl-connection). The connection returned is transient. The connection must be made from an existing stub to a block that would normally accept the manual addition of stubs.

Stubs are uniquely specified by a *symbol* and a *position*. The symbol can be a:

- **portname** - the name of the port at which the connection stub is attached.
- **side** - the top, bottom, left, or right side on which the stub exists.

The position, an integer, can be used to further specify which stub makes the connection. For example:

- If the symbol is a side and there are multiple connection stubs on that side, then the connection located at the specified position on that side is used.
- If the symbol is a portname and there are multiple stubs with that portname, then the stub located at the specified position, on any side, is used.

If the stub on *Obj1* is not uniquely specified (for example, a portname is given as *Sym1* without specifying a valid position and there are multiple connection stubs with the same portname), GDL signals an error. If *Obj2* is not a valid recipient of a gdl-connection (*Obj2* is a subclass of gdl-block and not a peer block, then gdl-make-connection-stub-to-block does not make a new connection, except for gdl-action-link.

If the inputs to gdl-make-connection-stub-to-block specify valid connection stubs, gdl-make-connection-stub-to-block makes the connection only if the stub on *Obj1* is not already connected. If the connection is not free, GDL signals an error. When GDL makes a connection, it returns the new connection. The connection is located on *Obj2* on the side specified by *Sym2* at the position specified by *Pos2*.

By passing a list of vertices to gdl-make-connection-stub-to-block, you can specify vertices on the new connection. To specify no vertices or to use the default vertices that G2 creates, pass an empty list as the *Vertices* argument.

Populate the *Vertices* list with integers that indicate the length of the connection before a bend occurs. To indicate the direction of the bend, use positive or negative integers. For more information, see the *G2 Reference Manual*.

Caution This procedure does not delete the list of vertices passed to it. Failure to delete the list after the procedure returns results in a leaked item.

The *InvalidateWorkspace* flag schedules the workspace of the newly created connection for recompilation, as described in gdl-compile-block-diagram. Proper initialization of gdl-connections occurs during the compilation process. For a

newly created connection to pass or display data, set the *InvalidateWorkspace* flag to true or make a separate call to gdl-compile-block-diagram.

Example

See "gdl-make-connection-block-to-block" on page 30.

gdl-make-connection-stub-to-stub

Creates a directed connection between existing stubs on two objects.

Synopsis

gdl-make-connection-stub-to-stub
(*Obj1*: class object, *Sym1*: symbol, *Pos1*: integer, *Obj2*: class object,
Sym2: symbol, *Pos2*: integer, *Vertices*: class integer-list,
InvalidateWorkspace: truth-value, *Client*: class object)
–> *Connection*: class gdl-connection

Arguments	Description
Obj1	The object on which to create a connection from an outward-directed stub.
Sym1	A symbol specifying either a portname or a side on *Obj1*.
Pos1	The position of the stub on *Obj1*.
Obj2	The object on which to create a connection to an inward-directed stub.
Sym2	A symbol specifying either a portname or a side on *Obj2*.
Pos2	The position of the stub on *Obj2*.
Vertices	An integer list of vertices that specify the bends of the connection as described in the *G2 Reference Manual*
InvalidateWorkspace	A value of true instructs the procedure to invalidate the workspace on which the connection was made.
Client	The g2-window or client object originating this call.

Return Value	Description
Connection	The newly created connection instance.

Description

This procedure creates a connection between two existing stubs on two specified objects, *Obj1* and *Obj2*. This procedure only works properly on connections that are used in GDL (subclasses of gdl-connection). The connection returned is transient and must be made between two existing stubs of the same class.

Stubs are uniquely specified by a *symbol* and a *position*. The symbol can be a:

- **portname** - the name of the port at which the connection stub is attached.
- **side** - the top, bottom, left, or right side on which the stub exists.

The position, an integer, can be used to further specify which stub makes the connection. For example, if the symbol is a side and there are multiple connection stubs on that side, then the connection located at the specified position on that side is used. If the symbol is a portname and there are multiple stubs with that portname, then the stub located at the specified position, on any side, is used.

If the stub on *Obj1* is not uniquely specified (for example, a portname is given as *Sym1* without specifying a valid position and there are multiple connection stubs with the same portname), GDL signals an error. If the stub on *Obj2* is not uniquely specified (for example, peer-input-blocks often have multiple inputs with the same portname), then a connection is made to one of the free connection ports.

If the inputs to gdl-make-connection-stub-to-stub specify valid connection stubs, gdl-make-connection-stub-to-stub makes the connection only if the connection stubs are not already connected and both stubs are of the same class. If either of these conditions are not met, GDL signals an error. When GDL makes a connection, it returns the new connection.

By passing a list of vertices to gdl-make-connection-stub-to-stub, you can specify vertices on the new connection. To specify no vertices or to use the default vertices that G2 creates, pass an empty list as the *Vertices* argument.

Populate the *Vertices* list with integers that indicate the length of the connection before a bend occurs. To indicate the direction of the bend, use positive or negative integers. For more information, see the *G2 Reference Manual*.

Caution This procedure does not delete the list of vertices passed to it. Failure to delete the list after the procedure returns results in a leaked item.

The *InvalidateWorkspace* flag schedules the workspace of the newly created connection for recompilation, as described in gdl-compile-block-diagram. Proper initialization of gdl-connections occurs during the compilation process. For a newly created connection to pass or display data, set the *InvalidateWorkspace* flag to true or make a separate call to gdl-compile-block-diagram.

Example

See “gdl-make-connection-block-to-block” on page 30.

gdl-reset-all-gdl-objects

Resets all objects.

Synopsis

gdl-reset-all-gdl-objects
(*Win*: class ui-client-item, *ResetPaths*: truth-value,
ShowProgress: truth-value)

Arguments	Description
Win	The ui-client-item originating this call.
ResetPaths	A value of true causes all blocks and paths to be reset.
ShowProgress	A value of true indicates that the progress bar is to be displayed.

Description

This procedure resets all gdl-objects in an application. It is equivalent to the reset all blocks menu choice.

Setting the value of *ResetPaths* to true resets gdl-paths and related data structures. Resetting causes existing values on paths to be eliminated. Setting the value of *ResetPaths* to true is equivalent to the Blocks and Paths option in the reset all blocks menu choice.

Setting the value of *ResetPaths* to false is equivalent to the Blocks Only option in the reset all blocks menu choice.

Setting the value of *ShowProgress* to true displays a progress bar on *Win* as the reset proceeds. If *Win* is the gfr-default-window then the progress bar appears on all windows. Setting *ShowProgress* to false results in no progress bar during reset.

gdl-reset-item

Resets an item, including blocks and paths.

Synopsis

gdl-reset-item
(*Item*: class item, *Client*: class object)

Arguments	Description
Item	The item to be reset.
Client	The g2-window or client object originating this call.

Description

This procedure is the equivalent of the reset menu choice on a block or a path. In addition, selecting reset on an object with a subworkspace resets all the items on the subworkspace of *Item*.

gdl-transfer-item

Transfers an item to a specified workspace at a specified location.

Synopsis

gdl-transfer-item
(*Item*: class item, *TargetWS*: class kb-workspace,
XPositionOnTarget: integer,*YPositionOnTarget*: integer,
InvalidateWorkspace: truth-value, *Client*: class object)

Arguments	Description
Item	The item to be transferred
TargetWS	The workspace to which the item is to be transferred
XPositionOnTarget	The X position, in workspace coordinates, to place the item.
YPositionOnTarget	The Y position, in workspace coordinates, to place the item.
InvalidateWorkspace	A value of true instructs the procedure to invalidate both the target and source workspaces.
Client	The g2-window or client object originating this call.

Description

This procedure transfers an item to a workspace at a specified location. The procedure is similar to the G2 transfer action: the item to be transferred must be transient and must not be connected to other items. Attempting to use this procedure to transfer an item illegally results in a signalled error.

The *InvalidateWorkspace* argument schedules both the original workspace of *Item* and the workspace *TargetWS* for recompilation, as described in "gdl-compile-block-diagram" on page 13. If the argument is set to false, no workspaces are invalidated.

Example

See "gdl-make-connection-block-to-block" on page 30.

gdl-turn-animation-off

Stops animation.

Synopsis

gdl-turn-animation-off
(*Client*: class object)

Argument	Description
Client	The g2-window or client object originating this call.

Description

This procedure turns off the animate option on the main menu. The gdl-turn-animation-off procedure asynchronously causes the global animation switch to be turned off. Animation is not turned off until all currently evaluating diagrams have completed execution.

gdl-turn-animation-on

Starts animation.

Synopsis

gdl-turn-animation-on
(*Client*: class object)

Argument	Description
Client	The g2-window or client originating this call.

Description

This procedure turns on the animate option on the main menu. The gdl-turn-animation-on procedure asynchronously causes the global animation switch to be turned on. Animation is not turned on until all currently evaluating diagrams have completed execution.

gdl-update-g2-chart

Updates the specified chart.

Synopsis

gdl-update-g2-chart
(*Chart*: class chart, *Client*: class object)

Arguments	Description
Chart	The chart to be updated.
Client	The g2-window or client object originating this call.

Description

This procedure is the equivalent of the update menu choice on a chart.

2

Custom Block Evaluator API

Describes classes and procedures used with the custom block evaluator.

Introduction

Three categories of procedures can be used with the custom block evaluator:

- Procedures that get values from a path.
- Procedures that propagate values onto a path and trigger the evaluation of downstream blocks.
- Procedures that set values onto a path without triggering the evaluation of downstream blocks.

In addition, two GDA functions resolve output path attributes for custom blocks.

The primary purpose of these procedures is to support the custom block evaluators, although these procedures can be called in other contexts. The propagate procedures, however, are only useful with custom block evaluators.

In GDA, we refer to values as being stored on paths. When you are working with a diagram, you can observe the values resident on a path by displaying a table. This table shows the attributes (e.g., value, collection time, timestamp) associated with that path. However, if you are writing procedures that access these values, it is critical to realize that these aren't attributes of the paths themselves.

For GDA, the attributes associated with paths are stored internally. This is done both for efficiency and to enable a network of paths to maintain a single path value, even when that network branches or crosses workspace boundaries. Access to these attributes is only through the APIs listed in this chapter. This includes access in the context of the block evaluators as well as simple programmatic access to the paths of a running diagram.

The synopsis and description for each procedure and function appear at the end of this chapter.

Using the Procedures that Get Values from a Path

These procedures get values from a path:

- gdl-get-data-path-value
- gdl-get-inference-path-value
- gdl-get-control-path-value
- gdl-get-path-item
- gdl-get-resident-path-item

These procedures get values for attributes associated with the path. See the *GDA User's Guide* for a description of the attributes associated with each path type.

The most common use of these procedures is to get values from an input path for a block. They can also be used to examine a value on an output path.

Note To reference paths by port name in the procedure, the port names must be unique within the class; i.e., not a peer input. You can reference peer inputs by path object, not port name. For more information, see the example on page 74.

Note If one of these procedures returns a negative timestamp, the port named in the call to the procedure does not exist for the block.

Using the Procedures that Set Values onto a Path

These categories of procedures set values onto a path:

- Procedures that set a value onto an output path and propagate the value to the downstream block, causing it to evaluate (these procedures are relevant only within a block evaluator).
- Procedures that set a value onto a path without causing the downstream block to evaluate.

Using the Procedures that Propagate Values Down a Path

These procedures propagate a value down a path and trigger the evaluation of the downstream block:

- gdl-propagate-data-path-value, described on page 93
- gdl-propagate-inference-path-value, described on page 94

- gdl-propagate-control-path-value, described on page 92
- gdl-propagate-path-item, described on page 96
- gdl-propagate-resident-path-item, described on page 97

Using the Procedures that Set Values Onto a Path

These procedures set a value onto a path without triggering the evaluation of the downstream block:

- gdl-set-data-path-value, described on page 99
- gdl-set-inference-path-value, described on page 100
- gdl-set-control-path-value, described on page 98
- gdl-set-path-item, described on page 101
- gdl-set-resident-path-item, described on page 97

Note To reference paths in the procedure, the port names must be unique.

Timestamps

A timestamp is a unique value that indicates the time that the value of a path was set. The timestamp is always a positive float that corresponds to a G2 time. You never set the timestamp of a path yourself; this is done automatically by G2.

Setting the Status Value for Inference Paths

When you set values onto an inference path using the gdl-propagate-inference-path-value or the gdl-set-inference-path-value procedure, the procedure enables you to specify the output belief value and status value. You can let GDA calculate the output status value based on the belief value and the output uncertainty of the block.

To cause GDA to calculate the status value based on the belief value, specify the symbol system as the status value argument to the procedure. By default, GDA assigns the following values for status-value based on the belief-value, when the Output Uncertainty of the block is 0.0:

The Status-value is...	When the Belief-value is...
.true	> 0.5
unknown	0.5
.false	< 0.5

If the block specifies a value for Output Uncertainty, GDA assigns the following values for Status-value based on the Belief-value and the Output Uncertainty (OU):

The Status-value is...	When the Belief-value is...
.true	≥ 0.5 + OU/2
unknown	between 0.5 - OU/2 and 0.5 + OU/2 (but not equal to either)
.false	≤ 0.5 - OU/2

If Output Uncertainty is not specified or is none, the block passes .true when the Belief-value ≥ 0.5 and .false when the Belief-value < 0.5; when Output Uncertainty is none, there is no Belief-value that results in a Status-value of unknown.

Using Item Paths

An **item path** is a path that carries not just a value, but any item. An **item** is any G2 item, which includes classes, lists, arrays, variables, parameters, displays, and text. Using an item path enables you to pass arbitrarily complex data between blocks.

No built-in GDA blocks use item paths. Therefore, to process items, you must build **custom blocks** that use item paths, described in the *GDA User's Guide*. Once you have created a custom block, you customize the block evaluator, also described in that manual.

When processing items within a custom block, you use one of two models:

- **The nonresident item model**, which processes items by passing each item from item path to block to item path to block in a diagram. In this method, items are not resident on the item path; once they are passed from the input path to the custom block, the input path is empty.
- **The resident item model**, which processes items resident on a path; each item is accessed by the block, but remains on the path after it is accessed. The block can change attributes on an item associated with a path but doesn't break the association.

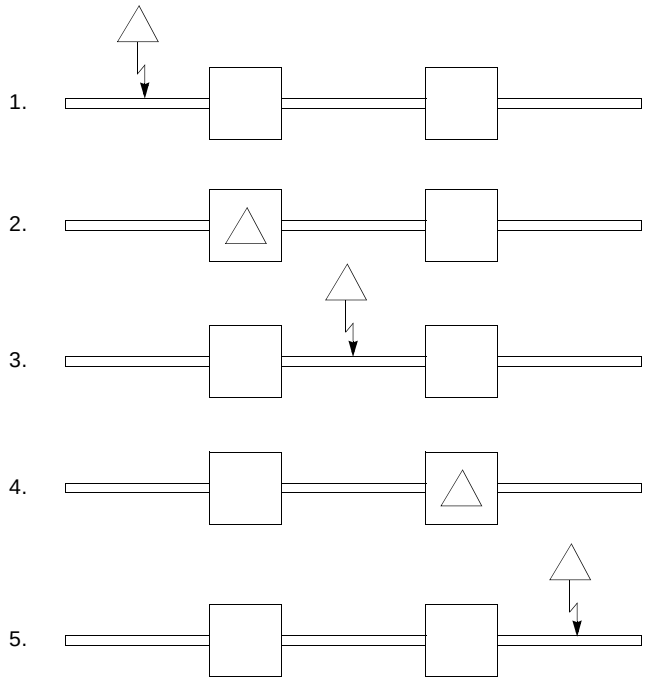
Depending on the desired model for processing items, the block evaluator uses different procedures to obtain an item from and place an item onto an item path. The following sections explain each of these two models in detail.

Using the Nonresident Item Model

This model uses the following procedures for obtaining items from and placing items onto paths. Note that an item path can only be associated with a single item at one time.

Procedure	Action	
gdl-get-path-item	**1**	Retrieves the item from the path for processing by the block.
gdl-set-path-item	**1**	If a transient item is on the path, it is deleted.
	2	Places the item on the path.
gdl-propagate-path-item	**1**	If a transient item is on the path, it is deleted.
	2	Places the item on the path.
	3	Triggers the evaluation of the downstream block.

The following sequence illustrates the nonresident item model. Each custom block has an input and an output item path. The triangles represent the items being processed. In this example, propagate procedures are used to place items on the output paths and trigger the evaluation of the downstream custom block.



The steps in the nonresident item model are:

1 An upstream component (not shown) supplies the item to the diagram, associating this item with the custom block's input item path and triggering the evaluation of the first block.

2 The block evaluator retrieves, then processes the item.

3 The block evaluator uses gdl-propagate-path-item to place the processed item onto the output item path. Because a propagate routine was used, the next block in the diagram is triggered and executes immediately.

4 The second custom block evaluator retrieves, then processes the item.

5 The block evaluator places the processed item onto the output item path.

In the nonresident item model, the item is passed from input path to output path by each custom block in the sequence.

Using the Resident Item Model

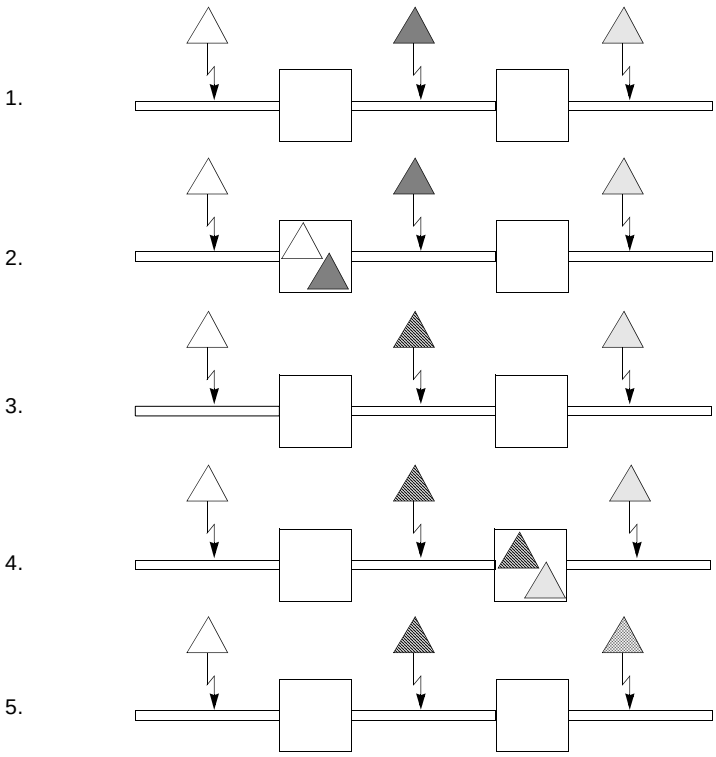
In the resident path model, the items remain "resident" on their paths. The blocks only change the data contained in these items and not the items themselves. They are analogous to the value paths normally associated with GDA blocks.

The difference between this method and the nonresident item method is that with the resident item method, the item resides on the path, while with the nonresident item method, the item moves through the diagram.

The resident item model uses the following procedures for obtaining items from and placing items onto paths.

Procedure	Action	
gdl-get-resident-path-item	**1**	Retrieves the item from a path for processing by the block. The item remains on the path.
gdl-set-resident-path-item	**1**	Informs GDA that a path item has changed.
gdl-propagate-resident-path-item	**1**	Informs GDA that a path item has changed.
	2	Triggers the evaluation of the downstream block.

The following diagram illustrates the resident item model. The first time a sequence of blocks evaluates using this model, each path is initialized, described in "Initializing Item Paths" on page 61. This diagram shows the blocks evaluating after initialization has taken place. Notice that in this example, the items do not move from the input and output item paths.



The steps in a resident item model are:

1. An upstream event or manual evaluation triggers the first custom block.
2. The custom block evaluator gets the item from the input path for processing. The block evaluator also gets the item from the output item path. The block evaluator then performs its operation, generating new values for the output item's attributes.
3. The block evaluator propagates the newly processed item onto the output path and triggers the evaluation of the downstream block. The item is the same item that was retrieved from the path, but the call to propagate causes the evaluation of the downstream blocks.
4. The second custom block evaluator gets the item from the input path for processing. The block evaluator also gets the item from its output path. The block evaluator then performs its operation, generating new values for the output item's attributes (as in step 2).
5. The block evaluator propagates the newly processed item onto the output path, continuing the sequence.

In the resident item model, the same item is resident on each path. The processing that takes place at each custom block simply modifies this item's attributes. The next item that comes to a custom block causes the block to evaluate, modifying attributes on the item on its output path.

Initializing Item Paths

The previous illustration assumes that all your item paths already have resident items associated with them. When your application first starts up, however, GDA has not associated items with each path. When gdl-get-resident-path-item is called for an item path that does not yet have an item associated with it, GDA initializes the path with an item of the correct class.

In developing your resident item path model, you have some class of item which must be present on the item paths in order for your model to work. To tell GDA what the appropriate class is for your item paths, you must create a function:

default-path-item-for-*class-name*

class-name in this case is the name of the class of the item paths. By default, item paths are of the class gdl-item-path. This function returns a symbol that names the class GDA is to instantiate to create the initial resident-path-item for your diagram. You can simply create the function default-path-item-for-gdl-item-path, or you can create custom classes of item paths by using "gdl-item-path" on page 79.

To define a function that initializes an item path:

1 Select KB Workspace > New Definition > function-definition.

2 Specify the function definition.

For example, suppose you want to use float-arrays as your resident path item. Suppose further that you have defined a new subclass of item path, which you have called array-path. In this way you can define different classes of item paths, each of which has its own initial item. Then you would define a function that initializes the resident path item for your subclass array to be a float-array. The function must return a symbol that names the class of item to create when a resident item path requires an initial item.

```
default-item-path-for-array-path() = the symbol FLOAT-ARRAY
```

Resolving Path Attributes

For custom blocks having multiple input paths, GDA provides two functions that resolve the quality and expiration of the custom block's output path: resolve-gdl-quality (described on page 107) and resolve-gdl-expiration (described on page 106).

When creating custom blocks that set values onto a data, inference, or control path, the block evaluator uses these functions. Custom blocks with item paths do not define path quality or expiration time, and custom blocks with control paths only define an output path quality.

For a general discussion of resolving output path attributes for peer input blocks, see the *GDA User's Guide*.

Classes

This section documents each of the GDA classes from which you can create a custom subclass.

gdl-block-with-evaluation-tracking

This class is a mix-in class. To use it you create a subclass that inherits from one of the custom block subclasses (gdl-custom-block, gdl-custom-peer-input-block, gdl-custom-multiple-invocation-block) and from this class.

This class adds an attribute to your class, time-of-last-evaluation, which stores the last time the block evaluated. This time can then be compared with the timestamps on input paths to determine whether a particular input arrived since the most recent evaluation.

This class adds resetting behavior so that when a block is reset, the time-of-last-evaluation is updated, so that path values that arrived before the reset are not counted as new. As with any custom block, if you extend the method gdl-reset, you must call next method so that the inherited behavior works properly.

Example

In this example, we create a new custom block that has any number of inference inputs, but only evaluates when all of the inputs have arrived. For simplicity, no analytical output is generated. We simply output a control signal.

Our subclass, wait-for-all-inputs, multiply inherits from gdl-custom-peer-input-block and gdl-block-with-evaluation-tracking. The following is the evaluator for this procedure (which has not used the Block Wizard to generate the skeleton).

```
wait-for-all-inputs-evaluator (Block: class wait-for-all-inputs, Mode: symbol,
    Client: class ui-client-item)
LastBlockEvaluation: float = the time-of-last-evaluation of Block;
TimeOfEvaluation: float;
InputPaths: class gdl-inference-path;
EachStatus, EachQuality, ResolvedOutputQuality: symbol;
EachBelief, EachCollectionTime, EachTimestamp: float;
EachExpiration: value;
AllPathsAreNew: truth-value = true;
begin
    ResolvedOutputQuality = Mode;
    for InputPaths = each gdl-inference-path connected at an input to Block
    do
        EachStatus, EachBelief, EachCollectionTime, EachQuality,
        EachExpiration, EachTimestamp = call gdl-get-inference-path-value
            (Block, InputPaths);
        if EachTimestamp > LastBlockEvaluation {Path is new} then
        ResolvedOutputQuality = resolve-gdl-quality (EachQuality,
            ResolvedOutputQuality)
        else AllPathsAreNew = false;
        exit if not (AllPathsAreNew); {Stop searching if an old path is found}
    end;
```

```
        if AllPathsAreNew then begin {Propagate output and update evaluation
        time}
            TimeOfEvaluation = call gdl-get-timestamp (Client);
            conclude that the time-of-last-evaluation of Block = TimeOfEvaluation;
            call gdl-propagate-control-path-value (Block, the symbol CP-OUT,
                ResolvedOutputQuality);
        end;
    end
```

Class Inheritance Path

gdl-block-with-evaluation-tracking, gdl-block, gdl-task, gdl-task-or-event, gdl-object, gfr-object-with-uuid, gdl-object-with-sse-id, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
time-of-last-evaluation	This read/write attribute is a slot to store the G2 time when a block last evaluated.
Allowable values:	Any float
Default value:	0.0
evaluator	See "gdl-custom-block" on page 66.
Allowable values:	Inherited
Default value:	GDL-DO-NOTHING

Extensible Methods

"gdl-object::gdl-clear-error" on page 110

"gdl-object::gdl-configure" on page 111

"gdl-object::gdl-get-configuration" on page 113

"gdl-object::gdl-initialize" on page 114

"gdl-object::gdl-reset" on page 115

gdl-custom-block

The General custom block specifies the basic set of attributes necessary for creating a custom GDA block. The General custom block performs evaluations on single or multiple inputs when the block needs to reference specific inputs by port name. The Difference block on the Arithmetic palette is similar to a General custom block because the bottom input is subtracted from the top input, which requires that the block evaluator refer to each input by name.

The General custom block can be subclassed to inherit the basic attributes and methods necessary to make instances of your subclasses function as blocks with a GDA diagram. When an instance of a custom block is connected as part of a GDA diagram, new data arriving causes the block's evaluator to be run.

The custom block evaluator is a procedure that takes three arguments. See "gdl-custom-block-evaluator" on page 109, for a general description.

A General custom block is one where the number of input and output ports is predefined. To create a custom block where the number and names of the input ports are not pre-specified, use the Peer Input block.

Note To create a custom block where the block evaluator does not identify the inputs by name, use "gdl-custom-peer-input-block" on page 73. To create a general purpose custom block that handles multiple simultaneous values, use "gdl-custom-multiple-invocation-block" on page 68.

For general information on how to specify the block and its associated procedure using the wizard, see the *GDA User's Guide*.

Class Inheritance Path

gdl-custom-block, gdl-custom-block-parent, gdl-block, gdl-task, gdl-task-or-event, gdl-object, gfr-object-with-uuid, gdl-object-with-sse-id, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
evaluator	A symbol that names the procedure that is executed when the block is evaluated.
Allowable values:	Any symbol
Default value:	GDL-DO-NOTHING

Example

For an example of a General custom block with one input path and one output path used to cube its input value, see "Creating a New Custom Subclass," and "Editing the Custom Portion of the Block Evaluator," in the *GDA User's Guide*.

Extensible Methods

"gdl-object::gdl-clear-error" on page 110

"gdl-object::gdl-configure" on page 111

"gdl-object::gdl-get-configuration" on page 113

"gdl-object::gdl-initialize" on page 114

"gdl-object::gdl-reset" on page 115

gdl-custom-multiple-invocation-block

A Multiple Invocation block is a block that, in G2 parlance, "allows other processing." Normally, GDA runs all blocks synchronously with data passing from block to block, and each block processing that data in turn. In some cases, however, a block may need to wait for an event or some period of time before producing an output and continuing the evaluation thread. For such blocks it also becomes necessary to determine what is to be done if additional data comes into the block while it is still processing the original data. It is also possible for a synchronous control block to receive multiple, simultaneous inputs. If you are building a custom block with this behavior, you must use the Multiple Invocation block as the parent.

The behavior of a Multiple Invocation block is controlled by two attributes. If you have defined your subclass using the Block Wizard, these attributes appear on the configuration dialog of the block. In many cases, however, these attributes would be set for the class, and not be configurable at each instance.

The asynchronous-evaluation attribute determines whether the block evaluates synchronously or asynchronously. In other words, it determines whether other GDA processing can take place while the evaluator of the block is in the process of running. Typically if you have any statements like wait for, collect data, or allow other processing (collectively referred to as a "wait state") or calls to other procedures that do the same, you would want to specify the asynchronous-evaluation attribute as true for your class.

Note Because the asynchronous-evaluation attribute is tied to the structure of the evaluator, you generally would not leave it configurable for the user of your custom block.

The multiple-invocations attribute determines how a block processes multiple signals received simultaneously, or additional data received while the block is evaluating. The options for this attribute are Ignore, OK, and Queue.

- When multiple-invocations is Ignore, any additional data that arrives at a block either simultaneously with other data or while a block is evaluating is ignored. That is, no additional evaluations take place using the new data.
- When multiple-invocations is OK, new arrival of data launches a new evaluation of the block, even if there is an evaluation in progress. This would generally be used with an asynchronous block, and involves data being processed in a diagram while the asychronous block is involved in a wait state. If a block is synchronous, there is no difference in behavior between OK and Queue.
- When multiple-invocations is Queue, each new arrival of data causes a new evaluation, but that evaluation only begins after any current evaluation has

completed. Additional data or simultaneous data thus "queues" for sequential processing.

Note A GDA path can only hold one data value at a time. Block evaluators, therefore, can only process the current data residing on the path, even if other data values actually caused the evaluation. For this reason, synchronous, multiple evaluation blocks or blocks whose multiple-invocations is set to true generally only make sense when used to process control path inputs.

The evaluator of a Multiple Invocation block looks much like the evaluator for any other custom block (see "gdl-custom-block-evaluator" on page 109). As with any other block, a propagate procedure is used to pass the data to an output path and invoke the downstream block. However, because the Multiple Invocation block can allow evaluation of other blocks as the evaluator is executing, it is conceivable that one would want to cause downstream blocks to evaluate during the evaluation procedure.

To cause a block to propagate data during a wait state, it is necessary to call the API procedure "gdl-trigger-evaluation-of-next-blocks" on page 105. It is not necessary to call the procedure as the evaluator is exiting. The following code fragment demonstrates how to output a control signal every 10 seconds and then again when the block evaluation is complete.

```
repeat
    wait for 10 seconds;
    RunningTotal = RunningTotal + 10;
    exit if RunningTotal <= the total-wait of MultipleInvocationBlock;
    call gdl-propagate-control-path-value (MultipleInvocationBlock,
        the symbol CP-OUT, OutputQuality);
    call gdl-trigger-evaluation-of-next-blocks (MultipleInvocationBlock, Mode,
        Client);
end;
call gdl-propagate-control-path-value (MultipleInvocationBlock,
    the symbol CP-OUT, OutputQuality);
{No trigger call necessary}
```

Class Inheritance Path

gdl-custom-multiple-invocation-block, gdl-custom-multiple-invocation-block-parent, gdl-asynchronous-block, gdl-block-with-invocation-management, gdl-block, gdl-task, gdl-task-or-event, gdl-object, gfr-object-with-uuid, gdl-object-with-sse-id, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
multiple-invocations	A symbol indicating how to handle multiple invocations.
Allowable values:	OK, QUEUE, IGNORE
Default value:	IGNORE
asynchronous-evaluation	A truth-value indicating whether the block evaluates asynchronously.
Allowable values:	Inherited
Default value:	true
evaluator	See "gdl-custom-block" on page 66.
Allowable values:	Inherited
Default value:	GDL-DO-NOTHING

Extensible Methods

"gdl-object::gdl-clear-error" on page 110

"gdl-object::gdl-configure" on page 111

"gdl-object::gdl-get-configuration" on page 113

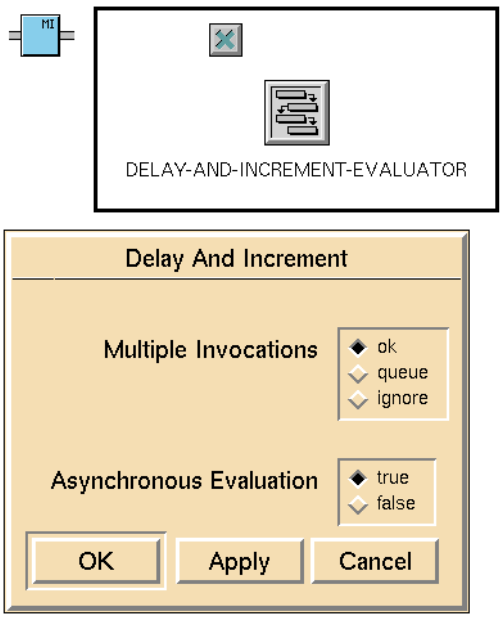
"gdl-object::gdl-initialize" on page 114

"gdl-object::gdl-reset" on page 115

Example

Suppose you want to create a block that delays processing its input signal by three seconds and adds 1 to the delayed input. To do this, create a Multiple Invocation custom block with a single input data path and a single output data path. To allow processing of multiple simultaneous inputs, specify multiple-invocations as OK. To allow other processing during the three-second delay, specify asynchronous-evaluation as true.

The following figure shows the block, the evaluator, the associated table, and the configuration dialog:



a delay-and-increment	
Notes	OK
Error	""
Comments	none
Invocations waiting	0
Invocations running	0

The following figure shows the Multiple Invocation custom block evaluator, and labels the custom portion of the code, which delays processing for three seconds, then increments the output value:

```
delay-and-increment-evaluator(blk: class delay-and-increment, mode: symbol, win: class object)
out-quality: symbol = mode; {the output quality}  out-expiration: value = the symbol none; {the output expiration time}
    out-collection:float = 0.0; {the output collection-time}
input-data-path-1-value:value; input-data-path-1-collection, input-data-path-1-timestamp:float; input-data-path-1-
    quality:symbol; input-data-path-1-expiration: value; {for data port input-data-path-1}
output-data-path-1-value:value; {for data port output-data-path-1}
begin

{Get input from data port input-data-path-1}
    input-data-path-1-value, input-data-path-1-collection, input-data-path-1-quality, input-data-path-1-expiration, input-
        data-path-1-timestamp = call gdl-get-data-path-value(blk,the symbol input-data-path-1);
    out-quality = resolve-gdl-quality(input-data-path-1-quality,out-quality);
    out-expiration = resolve-gdl-expiration(input-data-path-1-expiration,out-expiration);
    out-collection = max(input-data-path-1-collection,out-collection);

    {** Your code goes here **}

    wait for 3 seconds;
    output-data-path-1-value = input-data-path-1-value + 1;
```

Custom portion of procedure

```
{Set value for output data port output-data-path-1}
    call gdl-propagate-data-path-value(blk,the symbol output-data-path-1, output-data-path-1-value, out-collection,
        out-quality, out-expiration);
```

The following table shows sample input and output values for this block. The input and output values update once a second. The output values increment the input value associated with t_{n-3} by 1.

At time...	The input value is...	And the output value is...
0 seconds	0.0	no-value
1 second	1.0	no-value
2 seconds	2.0	no-value
3 seconds	3.0	1.0
4 seconds	4.0	2.0
5 seconds	5.0	3.0
6 seconds	6.0	4.0
7 seconds	7.0	5.0
8 seconds	8.0	6.0
9 seconds	9.0	7.0

gdl-custom-peer-input-block

The Peer Input custom block performs evaluations when the block treats all inputs the same. The Summation block on the Arithmetic palette is similar to a Peer Input block because all inputs are added together without requiring that they be identified individually. A Peer Input custom block specifies additional attributes that determine which inputs the block uses in its calculation based on the input quality.

For general information on peer input blocks, see the *GDA User's Guide*.

Note To create a custom block where the block evaluator identifies each input by its port name, use the "gdl-custom-block" on page 66 or the "gdl-custom-multiple-invocation-block" on page 68.

Creating Connections for Custom Peer Input Blocks

You can use the Custom Class Wizard to customize the connection ports of subclasses of the gdl-custom-peer-input-block class. When adding input ports to a Peer Input custom block, notice that, by default, the wizard does not name the port. This is because a Peer Input custom block treats all inputs equally and the block evaluator loops over all the inputs to determine their values. The wizard does, however, name all output ports of a Peer Input custom block.

Tip You must only create a single input port on a Peer Input custom block when you define the block. You drag additional input paths into an instance of the block on a workspace to create additional inputs. When creating the subclass, however, always create at least one input port to generate the correct block evaluator.

Customizing the Peer Input Custom Block Evaluator

When writing an evaluator for a Peer Input block, instead of making individual API calls to get the input values, the inputs are collected in a for-loop. If you use the Custom Class Wizard to construct your Peer Input custom block, the procedure is built for you. In this case, a single local variable is defined, which is used to loop over all paths connected at an input to the block. Additionally, single local variable names are defined for each of the path attributes (see the example for more details). This is in contrast to a General custom block where each path is accessed separately, and the Wizard generates individual local names for each attribute of each path.

Because of the way the default Peer Input block evaluator is designed, several things follow:

- The block evaluator for a Peer Input custom block cannot control the order of evaluation of the inputs.
- The custom portion of the block evaluator cannot differentiate among individual input path values for paths of the same type, e.g., data paths.
- You can drag input paths into the block on a workspace after you customize the subclass.
- You do not need to name input paths when adding connections to the block using Stub Tools or dragging connections into the block on a workspace.

Extensible Methods

"gdl-object::gdl-clear-error" on page 110

"gdl-object::gdl-configure" on page 111

"gdl-object::gdl-get-configuration" on page 113

"gdl-object::gdl-initialize" on page 114

"gdl-object::gdl-reset" on page 115

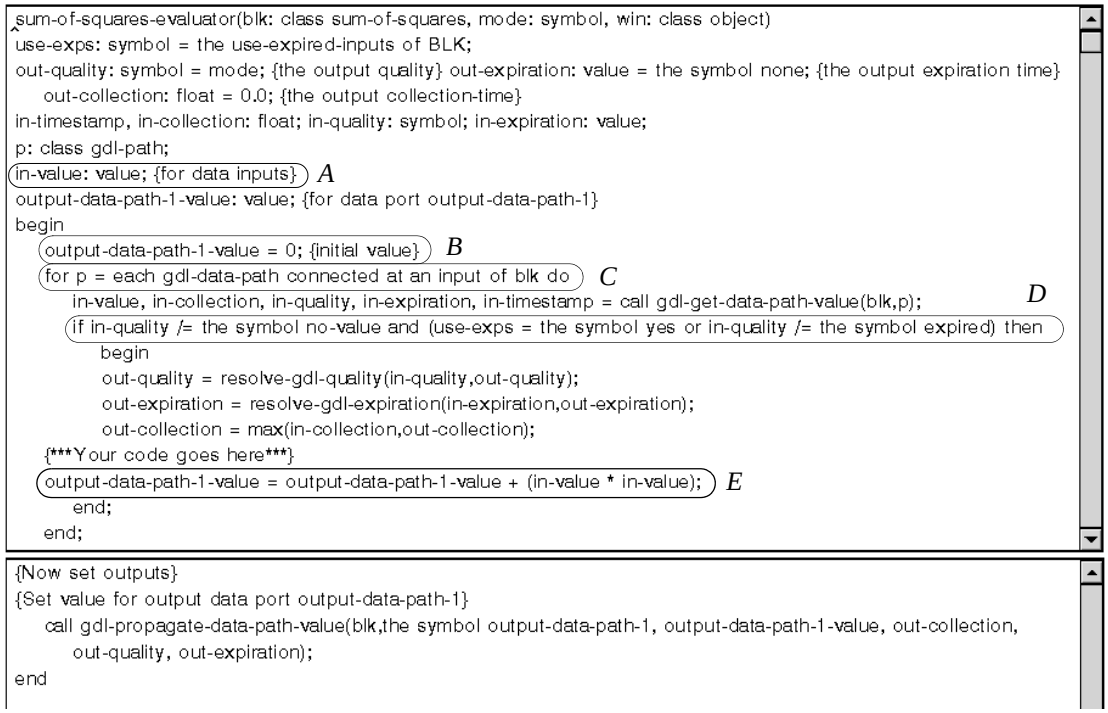
Example

Suppose you want to create a custom block that calculates the sum of the squares of its inputs. To do this, use the Custom Class Wizard to create a Peer Input custom block that sets the output path value to the square of its input values. The default block evaluator loops over all of the inputs, which processes all of the input paths, regardless of their names or number.

The following figure shows the Peer Input custom block, its evaluator, and its table. Notice that the block only has one input path by default. You can edit the value of the Use-expired-inputs attribute in the table.

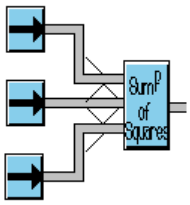
a sum-of-squares	
Notes	OK
Error	""
Comments	none
Use expired inputs	yes

The figure labels each portion of the block evaluator. The table following the figure provides a brief explanation of each labeled portion of the block evaluator.



The portion labeled...	Does this...
A	Declares a single local name for all input paths
B	Initializes the output path
C	Defines the for-loop that loops over input paths to obtain their value
D	Determines output path quality
E	The custom portion of the procedure, which sets the output value to the sum of the squares of each input value

The following figure illustrates how you use this block to sum the squares of a custom block with three input paths. You drag the input paths into the block on a workspace after you create the custom block. The input values are 2, 3, and 4, and the output path Data-value is 29.



a gdl-data-path	
Quality	manual
Timestamp	7652.0
Collection time	7652.0
Expiration time	none
Data value	29.0

gdl-dialog-intermediary

The GDL Dialog Intermediary class, or GDI, contains the configuration information for blocks and other GDA classes. A GDI gives the dialog, a display name, and the public attributes for the class which it represents. Although a GDI exists for most block classes within GDA, you probably only access or modify the attributes of a GDI if you create a custom class and need to control its configurability. If you create your custom class using the Custom Block Wizard, a GDI is automatically created for you and stored with the class definition. If you are creating your own subclass of a custom class and want to control certain configuration options, you have to create a GDI programmatically, as the class does not appear on the GDA palettes.

Three pieces of information are stored in the GDI:

- The name of the UIL dialog to be used when the associated block is configured from a G2 window.
- A "common name" for the class, which is used on the dialog labels (both GUIDE and native) when a name for the block being configured does not exist.
- A list of public attributes that are retrieved by the API gdl-get-configuration, and are the attributes you can control using the Set Attribute block.

The GDI is a subclass of symbol-array. The list of public attributes is stored as the elements of the GDI itself.

Class Inheritance Path

gdl-dialog-intermediary, symbol-array, value-array, g2-array, object, item

Attributes

Attribute	Description
id	The class to which this intermediary applies.
Allowable values:	Any symbol
Default value:	GDI-DEFAULT

Attribute	Description
dialog	Either the symbol gdl-no-configurable-attributes, the symbol gdl-configure-on-subws, or a text that corresponds to the id of some UIL dialog.
Allowable values:	Any item or value
Default value:	NONE
common-name	A text description of the class named in the id, used to label the dialog, if the instance being configured is not named.
Allowable values:	Any item or value
Default value:	NONE
user-data	This field is provided to contain additional data as needed.
Allowable values:	Any item or value
Default value:	NONE

gdl-item-path

You can create customized item path connections to pass particular types of data for different custom blocks.

If you want to create your own item path, you should subclass gdl-item-path. You generally would create your own subclass of gdl-item-path to customize the default item that is created to initialize a resident-item-path (see "Initializing Item Paths" on page 61). You can also customize the appearance of your item paths, as explained under the description for gdl-path, on page 81.

Customizing the Path Regions for Item Paths

gdl-item-path defines four regions: sides, center, light, and dark. When you define your own subclass, it should also include these four regions. Further, the "center" region should be prominent, as this is the region whose color is changed when a path is highlighted and unhighlighted. If you have changed the default color of the center region, you must also override the method gdl-unhighlight for your class.

Adding Custom Connections to a Custom Subclass

You cannot add custom connections to a custom subclass using the wizard, as you can with GDA-defined classes. To add custom connections to a custom subclass block, you must edit the stubs attribute of the class definition for your custom block. See the *G2 Reference Manual* for details of how to modify connections for an object class.

Class Inheritance Path

gdl-item-path, gdl-path-without-junctions, gdl-path, gdl-connection, connection, item

Attributes

none

Extensible Methods

gdl-item-path::gdl-unhighlight

gdl-object

This is the base class for most of the objects in GDA. It should not be directly subclassed or extended. It is included here for reference only.

Class Inheritance Path

gdl-object, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
error	If an error occurs on a block or other object, the text of the error appears in this attribute. This attribute is read-only.
Allowable values:	Any text
Default value:	""
comments	This attribute is not used by GDA. It can be used for any purpose desired by GDA users.
Allowable values:	Any item or value
Default value:	NONE

gdl-path

You can subclass gdl-path or its subclasses (gdl-data-path, gdl-inference-path, gdl-action-path, and gdl-item-path) to modify the appearance of the paths on your diagrams. In addition, when subclassing gdl-item-path, you can modify the behavior of those paths (as described for that class).

Customizing the Connection Path Regions

Subclasses of gdl-path define two regions: sides and center. When you define your own subclass, it should also include these regions. The center region is the one whose color is changed when a path is highlighted and unhighlighted. If, in your subclass, you have changed the default color of the center region, you must also override the method gdl-unhighlight for your class.

Additionally, the subclass gdl-data-path defines a region "type," which should also be included in your definition.

By editing the cross-section-pattern of your subclass, you can override either the colors, the thickness, or both for the connections used by your custom blocks.

For example, to define a subclass of data path, which has a default color of white (rather than gray) and defines an addition region, alarm, take the following steps.

To create a new connection subclass

1. Create a new connection-definition.
2. Specify the direct-superior-classes to be gdl-data-path.
3. Specify a unique symbol for class-name.
4. Modify the cross-section-pattern to contain your new regions and specifications, but you must still define the GDA-defined regions.

   ```
   sides=black, center=white, type=gray, alarm=red;
   1 sides, 2 center, 2 alarm, 2 center, 1 sides
   ```

   Notice that it was not necessary to use the GDA-defined regions in your specification, but it was necessary to define them.

Adding Custom Connections to a Custom Subclass

You cannot add custom connections to a custom subclass using the wizard, as you can with GDA-defined classes. To add custom connections to a custom subclass block, you must edit the stubs attribute of the class definition for your custom block. See the *G2 Reference Manual* for details of how to modify connections for an object class.

Class Inheritance Path

gdl-path, gdl-connection, connection, item

Attributes

none

gdl-simple-encapsulation

This is the parent class of any user-defined encapsulation block. Although you would normally use instances of the class gdl-encapsulation-block (found on the GDA palettes), you must use this class when creating your own subclasses of encapsulations. For example, you may want to create subclasses of encapsulation blocks if you want to customize the icon or add custom attributes. Note that even when you create a subclass on gdl-simple-encapsulations, the diagrams of instances of that class can vary from instance to instance. If you want to standardize the behavior of an encapsulation across all instances of a class, use the Single Source Encapsulation block (gdl-single-source-encapsulation).

Existing attributes of this class or subclasses, including the evaluator, should not be modified. Rather than creating a custom evaluator, you customize the behavior of an encapsulation block by modifying the diagram on its subworkspace. The *GDA User's Guide* contains more information on the use of encapsulation blocks.

Although subclasses of gdl-simple-encapsulation can be created directly, you would generally use the Custom Class Wizard, described in the *GDA User's Guide*.

Class Inheritance Path

gdl-simple-encapsulation, gdl-encapsulation-block, gdl-block, gdl-task, gdl-task-or-event, gdl-object, gfr-object-with-uuid, gdl-object-with-sse-id, object, gfr-item-with-uuid, item

Attributes

none

Extensible Methods

"gdl-object::gdl-clear-error" on page 110

"gdl-object::gdl-configure" on page 111

"gdl-object::gdl-get-configuration" on page 113

"gdl-object::gdl-initialize" on page 114

"gdl-object::gdl-reset" on page 115

gdl-single-source-encapsulation

This is the parent class of any user-defined single-source-encapsulation (SSE). You do not use instances of gdl-single-source-encapsulation directly; you must create your own subclass.

SSE's behave in the same way as standard encapsulation blocks, except that each subclass of SSE has associated with it a Master Diagram. Editing the Master Diagram for a class modifies the diagrams for each individual instance of that class. For information on the use of any gdl-single-source-encapsulation subclass, see the SSE, described in the *GDA Reference Manual*. Like subclasses of the standard encapsulation (gdl-simple-encapsulation), the existing attributes, including the evaluator, should not be modified. Instead, you customize the behavior of an encapsulation block by modifying the diagram on its subworkspace.

Although subclasses of gdl-single-source-encapsulation can be created directly, you would generally use either the Custom Class Wizard, described in the *GDA User's Guide*, or the convert-to-sse menu choice.

Class Inheritance Path

gdl-single-source-encapsulation, gdl-single-source-encapsulation-parent, gdl-encapsulation-block, gdl-block, gdl-task, gdl-task-or-event, gdl-object, gfr-object-with-uuid, gdl-object-with-sse-id, gcc-object-with-destructor, gcc-object, object, gfr-item-with-uuid, item

Attributes

none

Extensible Methods

"gdl-object::gdl-clear-error" on page 110

"gdl-object::gdl-configure" on page 111

"gdl-object::gdl-get-configuration" on page 113

"gdl-object::gdl-initialize" on page 114

"gdl-object::gdl-reset" on page 115

Procedures and Functions

This section describes the procedures and functions useful with the custom block evaluator.

gdl-get-control-path-value

Gets the timestamp and quality from a control path.

Synopsis

gdl-get-control-path-value
(*Block*: class object, *PathReference*: item-or-value)
-> (*float*, *symbol*)

Argument	Description
Block	The object to which the data path is connected. Typically of class gdl-block, but could also be a subclass of variable-or-parameter. When used with a custom block evaluator, this is the block being evaluated by the evaluator, an existing class of custom block.
PathReference	A path attached to an input or output of *Block*, a path of class gdl-control-path or a symbol that refers to the portname.

Return Value	Description
float	The timestamp of the path.
symbol	The quality of the path, either OK, Manual, Expired, and No-value.

gdl-get-data-path-value

Gets the data value, collection time, quality, expiration time, and timestamp from a data path.

Synopsis

gdl-get-data-path-value
(*Block*: class object, *PathReference*: item-or-value)
–> (*value*, *float*, *symbol*, *value*, *float*)

Argument	Description
Block	The object to which the data path is connected. Typically of class gdl-block, but could also be a subclass of variable-or-parameter. When used with a custom block evaluator, this is the block being evaluated by the evaluator, an existing class of custom block. To use the example from the Custom Block Wizard chapter of the *GDA User's Guide*, power-block.
PathReference	A path attached to an input or output of *Block*. This must either be a subclass of gdl-data-path or a symbol that refers to a portname.

Return Value	Description
value	The value on the path.
float	The collection time associated with the data value. Typically, the collection time is the time the data entered GDA.
symbol	The quality of the path. Allowable values are OK, Manual, Expired, and No-value.
value	The expiration time of the data associated with the path.
float	The timestamp of the path.

gdl-get-inference-path-value

Gets the status and belief values, collection time, quality, expiration time, and timestamp from an inference path.

Synopsis

gdl-get-inference-path-value
(*Block*: class object, *PathReference*: item-or-value)
–> (*symbol*, *float*, *float*, *symbol*, *value*, *float*)

Argument	Description
Block	The object to which the data path is connected. Typically of class gdl-block, but could also be a subclass of variable-or-parameter. When used with a custom block evaluator, this is the block being evaluated by the evaluator, an existing class of custom block.
PathReference	A path attached to an input or output of *Block*, a path of class gdl-inference-path or a symbol that refers to the portname.

Return Value	Description
symbol	The status of the path.
float	The belief value on the path.
float	The collection time of the data associated with the path.
symbol	The quality of the path. Allowable values are OK, Manual, Expired, and No-value.
value	The expiration time of the data associated with the path.
float	The timestamp of the path.

gdl-get-path-item

Gets the item from an item path.

Synopsis

gdl-get-path-item
(*Block*: class object, *PathReference*: item-or-value)
–> (*item-or-value*)

Argument	Description
Block	The object to which the data path is connected. Typically of class gdl-block, but could also be a subclass of variable-or-parameter. When used with a custom block evaluator, this is the block being evaluated by the evaluator, an existing class of custom block.
PathReference	A path attached to an input or output of *Block*, a path of class gdl-item-path or a symbol that refers to the portname.

Return Value	Description
item-or-value	The item found on the path, or the symbol NO-ITEM if no item was found on the path.

Description

This routine is similar to gdl-get-resident-path-item but it removes the item from the path. It also does not create default path items if the path is empty. It returns NO-ITEM if the path is empty or if the port does not exist.

For information about the difference between this procedure and gdl-get-resident-path-item, see "Using Item Paths" on page 57.

gdl-get-resident-path-item

Gets the item from an item path.

Synopsis

gdl-get-resident-path-item
(*Block*: class object, *PathReference*: item-or-value)
–> (*item-or-value*)

Argument	Description
Block	The object to which the data path is connected. Typically of class gdl-block, but could also be a subclass of variable-or-parameter. When used with a custom block evaluator, this is the block being evaluated by the evaluator, an existing class of custom block.
PathReference	A path attached to an input or output of *Block*, a path of class gdl-item-path or a symbol that refers to the portname.

Return Value	Description
item-or-value	The item found on the path. If no item is found, the default item is generated. If no function can be located or the path specified does not exist, the symbol NO-ITEM is returned.

Description

This routine is similar to gdl-get-resident-path-item but it does not remove the item from the path. A default path item is created if the path is empty. It returns NO-ITEM if the port does not exist.

For information about the difference between this procedure and gdl-get-path-item, see "Using Item Paths" on page 57.

gdl-get-timestamp

Returns a unique timestamp in standard G2 time.

Synopsis

gdl-get-timestamp
(*Client*: ui-client-item)
–> (*float*)

Argument	Description
Client	The client for the call.

Return Value	Description
float	A timestamp, in G2 time.

Description

This routine returns a unique timestamp, in G2 time. This is roughly equivalent to the G2 expression the current subsecond time, which gives the number of seconds since G2 was last started. However, this procedure assures that each value returned is unique, whether or not the G2 clock has advanced between calls to gdl-get-timestamp.

This procedure is used internally by GDA to generate the Timestamp value of a GDA path. Although it is not necessary for you to ever set the Timestamp of a path, the gdl-get-timestamp function is useful if you ever need to compare path Timestamps with the current time. For an example of proper usage, see "gdl-block-with-evaluation-tracking" on page 64.

gdl-propagate-control-path-value

Propagates a "token" or "impulse" down a control path.

Synopsis

gdl-propagate-control-path-value
(*Block*: class object, *PathReference*: item-or-value, *Quality*: symbol)

Argument	Description
Block	The block being evaluated by the calling procedure to which the path is attached. This would typically be an existing class of custom block.
PathReference	A path attached to the output of *Block*, a path of class gdl-control-path or a symbol that refers to the portname; for example, the symbol CP-OUT.
Quality	The path quality.

Description

Propagates a token or impulse down a control path. Propagating a control path does not send any values, but simply causes the next block in the diagram to evaluate. Control paths also have a quality associated with them, which must be included in the propagate call.

gdl-propagate-data-path-value

Propagates the data value down a data path.

Synopsis

gdl-propagate-data-path-value
(*Block*: class object, *PathReference*: item-or-value, *Value*: value,
CollectTime: float, *Quality*: symbol, *Expiration*: value)

Argument	Description
Block	The block whose evaluator is propagating the data. This would typically be an existing class of custom block. To use the example from the Custom Block Wizard chapter of the *GDA User's Guide*, power-block.
PathReference	A path attached to the output of *Block* (a subclass of gdl-data-path) or a symbol that refers to the portname. For example, the symbol DP-OUT.
Value	The data value.
CollectTime	The collection time for the originating data. This would typically be the collection time from the input path.
Quality	The quality, typically taken from the input paths. See "Resolving Path Attributes" on page 62.
Expiration	The expiration time. See "Resolving Path Attributes" on page 62.

Description

Propagates the data value, along with its associated collection time, quality, and expiration time, down a data path. Typically, the data value is a quantity, but a data path can carry any G2 value type.

gdl-propagate-inference-path-value

Propagates an output data value down an inference path.

Synopsis

gdl-propagate-inference-path-value
(*Block*: class object, *PathReference*: item-or-value, *Status*: symbol, *Belief*: float, *CollectTime*: float, *Quality*: symbol, *Expiration*: value)
–> (*symbol*)

Argument	Description
Block	The block whose evaluator is being called. This would typically be an existing class of custom block.
PathReference	A path attached to the output of *Block*, a path of class gdl-inference-path or a symbol that refers to the portname; for example, the symbol IP-OUT.
Status	The status (.true, .false, or unknown) or the symbol SYSTEM.
Belief	The belief value, a float between 0.0 and 1.0.
CollectTime	The collection time for the originating data, typically the collection time from the input path.
Quality	The quality, typically taken from the input paths. See "Resolving Path Attributes" on page 62.
Expiration	The expiration time. See "Resolving Path Attributes" on page 62.

Return Value	Description
symbol	The status. If SYSTEM is specified, the computed status is returned.

Description

This procedure propagates the status and/or belief values, including the collection time, quality, and expiration time associated with those values, down an inference path. Propagation triggers the evaluation of any downstream blocks.

Typically, the Belief and Status of a path are related. (See "Setting the Status Value for Inference Paths" on page 56 for more information.) When you propagate an inference path value, you have the option of determining the Status to be propagated yourself, or requesting GDA to make that calculation for you. You control this choice by specifying the Status in your call to this procedure.

- If Status is any of the symbols .TRUE, .FALSE, or UNKNOWN, the resulting status on the output path will be the status you specify.
- However, if Status is the symbol SYSTEM, then GDA will determine the status for you. You have access to the calculated status as the return value from this procedure.

gdl-propagate-path-item

Propagates the item down an item path.

Synopsis

gdl-propagate-path-item
(*Block*: class object, *PathReference*: item-or-value, *Item*: class item)

Argument	Description
Block	The block being evaluated by the calling procedure to which the path is attached. This would typically be an existing class of custom block.
PathReference	A path attached to the output of *Block*, a path of class gdl-item-path or a symbol that refers to the portname.
Item	An instance of any item subclass.

Description

This procedure propagates an output data value to all of the connected downstream blocks and causes those downstream blocks to become scheduled for immediate evaluation. It has the side effect of deleting the existing object associated with the output path, if that item is transient.

For more information about the difference between this procedure and gdl-propagate-resident-path-item, see "Using Item Paths" on page 57.

If *Item* has an attribute timestamp, the GDA-computed timestamp is concluded into this attribute.

gdl-propagate-resident-path-item

Propagates an output value to connected downstream blocks.

Synopsis

gdl-propagate-resident-path-item
(*Block*: class object, *PathReference*: item-or-value, *Item*: class item)

Argument	Description
Block	The block whose evaluator is propagating the item. This would typically be an existing class of custom block.
PathReference	A path attached to the output of *Block*, a path of class gdl-item-path or a symbol that refers to the portname.
Item	The item retrieved from the path using gdl-get-resident-path-item.

Description

This procedure propagates an output data value to all of the connected downstream blocks and causes those downstream blocks to become scheduled for immediate evaluation. If a different path item exists, the item is replaced, but the previous path item is not deleted, even if it is transient. Use gdl-propagate-path-item, instead, to delete the previous path item.

For information about the difference between this procedure and gdl-propagate-path-item, see "Using Item Paths" on page 57.

If *Item* has an attribute timestamp, the GDA-computed timestamp is concluded into this attribute.

gdl-set-control-path-value

Sets a control path.

Synopsis

gdl-set-control-path-value
(*Block*: class object, *PathReference*: item-or-value, *Quality*: symbol)

Argument	Description
Block	The object to which the data path is connected. Typically of class gdl-block, but could also be a subclass of variable-or-parameter. When used with a custom block evaluator, this is the block being evaluated by the evaluator, an existing class of custom block.
PathReference	A path attached to an input or output of *Block*, a path of class gdl-control-path or a symbol that refers to the portname.
Quality	The path quality.

Description

Sets the quality and a new timestamp onto a control path.

This routine is similar to gdl-propagate-control-path-value but does not trigger the evaluation of downstream blocks.

gdl-set-data-path-value

Sets the data value, collection time, quality, and expiration time onto a data path.

Synopsis

gdl-set-data-path-value
(*Block*: class object, *PathReference*: item-or-value, *Value*: value,
CollectTime: float, *Quality*: symbol, *Expiration*: value)

Argument	Description
Block	The object to which the data path is connected. Typically of class gdl-block, but could also be a subclass of variable-or-parameter. When used with a custom block evaluator, this is the block being evaluated by the evaluator, an existing class of custom block.
PathReference	A path attached to an input or output of *Block*. This must either be a subclass of gdl-data-path or a symbol that refers to a portname.
Value	The data value.
CollectTime	The collection time for the originating data. This would typically be the collection time from the input path.
Quality	The quality, typically taken from the input paths. See "resolve-gdl-quality" on page 107.
Expiration	The expiration time. See "resolve-gdl-expiration" on page 106.

Description

This procedure propagates an output data value to all of the connected downstream blocks and causes those downstream blocks to become scheduled for immediate evaluation. It has the side effect of deleting the existing object associated with the output path, if that item is transient.

gdl-set-inference-path-value

Sets the status and belief values, collection time, quality, and expiration time onto an inference path.

Synopsis

gdl-set-inference-path-value
(*Block*: class object, *PathReference*: item-or-value, *Status*: symbol, *Belief*: float, *CollectTime*: float, *Quality*: symbol, *Expiration*: value)
–> (*symbol*)

Argument	Description
Block	The object to which the data path is connected. Typically of class gdl-block, but could also be a subclass of variable-or-parameter. When used with a custom block evaluator, this is the block being evaluated by the evaluator, an existing class of custom block.
PathReference	A path attached to an input or output of *Block*, a path of class gdl-inference-path or a symbol that refers to the portname.
Status	The status value.
Belief	The belief value.
CollectTime	The collection time.
Quality	The quality.
Expiration	The expiration time.

Return Value	Description
symbol	The status.

Description

This procedure propagates an output data value to all of the connected downstream blocks and causes those downstream blocks to become scheduled for immediate evaluation. It has the side effect of deleting the existing object associated with the output path, if that item is transient.

gdl-set-path-item

Sets an item on an item path.

Synopsis

gdl-set-path-item
(*Block*: class object, *PathReference*: item-or-value, *Item*: class item)

Argument	Description
Block	The object to which the data path is connected. Typically of class gdl-block, but could also be a subclass of variable-or-parameter. When used with a custom block evaluator, this is the block being evaluated by the evaluator, an existing class of custom block.
PathReference	A path attached to an input or output of *Block*, a path of class gdl-item-path or a symbol that refers to the portname.
Item	An instance of any item class.

Description

Sets the item onto an item path. This routine is similar to gdl-propagate-path-item except that no downstream evaluation is triggered. For information about the difference between this procedure and gdl-propagate-path-item, see "Using Item Paths" on page 57.

If *Item* has an attribute timestamp, the GDA-computed timestamp is concluded into this attribute.

gdl-set-resident-path-item

Synopsis

gdl-propagate-resident-path-item
(*Block*: class object, *PathReference*: item-or-value, *Item*: class item)

Argument	Description
Block	The object to which the data path is connected. Typically of class gdl-block, but could also be a subclass of variable-or-parameter. When used with a custom block evaluator, this is the block being evaluated by the evaluator, an existing class of custom block.
PathReference	A path attached to an input or output of *Block*, a path of class gdl-item-path or a symbol that refers to the portname.
Item	An instance of any item class.

Description

Propagates the item down an item path. For information about the difference between this procedure and gdl-propagate-path-item, see "Using Item Paths" on page 57.

If *Item* has an attribute timestamp, the GDA-computed timestamp is concluded into this attribute.

gdl-trigger-asynchronous-evaluation

Propagates an output from outside the normal flow of evaluation when you are propagating asynchronously within a custom Multiple Invocation block.

Synopsis

gdl-trigger-asynchronous-evaluation
(*Blk*: class gdl-custom-multiple-invocation-block, *Mode*: symbol,
Client: class object)

Arguments	Description
Blk	The user-defined subclass of a Multiple Invocation block.
Mode	A symbol of manual indicates that the inputs have been supplied manually by using the override mode; a symbol of automatic indicates that the inputs result from a data-driven evaluation of the block.
Client	The g2-window or client object originating this call.

Description

This procedure propagates an output from outside the normal flow of evaluation, such as from repeat loops inside evaluations or from outputs following wait states. The gdl-trigger-asynchronous-evaluation procedure is called only within the evaluator of a custom Multiple Invocation block and only when the asynchronous-evaluation attribute of the block is set to true.

Note Use gdl-trigger-asynchronous-evaluation only when a block needs to propagate an output from outside the normal flow of evaluation.

Because a call to gdl-trigger-asynchronous-evaluation does not by itself cause block evaluation, the gdl-trigger-asynchronous-evaluation procedure must be preceded by a call to one of these API procedures that propagate values down a path:

- gdl-propagate-data-path-value
- gdl-propagate-inference-path-value
- gdl-propagate-control-path-value

These procedures set a value onto a path and propagate the value to downstream blocks. The gdl-trigger-asynchronous-evaluation procedure then causes them to evaluate.

In almost all cases, the arguments to gdl-trigger-asynchronous-evaluation are the same as those passed into the evaluator.

For more information on the creating custom Multiple Invocation blocks, see the *GDA User's Guide*.

gdl-trigger-evaluation-of-next-blocks

Triggers the blocks invoked in a previous call.

Synopsis

gdl-trigger-evaluation-of-next-blocks
(*Obj*: object, *Mode*: symbol, *Client*: ui-client-item)

Argument	Description
Obj	Typically, this must be a subclass of gdl-custom-multiple-invocation-block.
Mode	The mode passed into the evaluator that is calling this procedure.
Client	The client passed into the evaluator that is calling this procedure.

Description

This procedure should be called when propagating from outside the normal flow of evaluation, such as from repeat loops inside evaluations. This procedure does not invoke the blocks, it merely triggers the blocks that have been invoked in a previous call. Therefore, it is necessary to call gdl-propagate-xxx-path-value before the call to this procedure.

For any classes that are not Multiple Invocation blocks, this call is not necessary because any propagates remain in the normal flow of evaluation. Also, if the propagate call is made just before the evaluator returns, it is not necessary to call this procedure.

See “gdl-custom-multiple-invocation-block” on page 68 for an example of how to use this procedure from an evaluator.

resolve-gdl-expiration

Returns the minimum of two expiration times.

Synopsis

resolve-gdl-expiration
(*Expiration-1*, *Expiration-2*)
–> (*value*)

Argument	Description
Expiration-1	
Expiration-2	

Return Value	Description
value	The minimum of the *Expiration-1* and *Expiration-2*.

Description

Returns the minimum of two expiration times.

resolve-gdl-quality

Returns the lower of two qualities.

Synopsis

resolve-gdl-quality
(*Quality-1*, *Quality-2*)
–> (*symbol*)

Argument	Description
Quality-1	
Quality-2	

Return Value	Description
symbol	The lower of *Quality-1* and *Quality-2*.

Description

Returns the lower of two qualities.

Extensible Methods

GDA provides extensible classes as part of the Custom Class Wizard. Using the wizard, you can customize the appearance and attributes of custom blocks. By writing an evaluator procedure, you can control the behavior of custom blocks when evaluated. If necessary, you can specify additional actions that the block takes on startup, reset, or when in error. For most custom blocks, it is unnecessary to write any specific methods.

For each of these methods, the argument *Client* is the window of the user who initiated the action, or the object gfr-default-window, in the case of system initiated calls. *Obj* is the gdl-object being acted upon.

When you write your own extensions to these methods, you must insert the name of your customized subclass in the declaration instead of the class gdl-object. You must also include a call next method in any of these method specifications to assure that the functionality of the superior classes is inherited properly.

Caution Do not call any of these methods directly in any of your code. You should instead use the API functions described in this manual. Attempts to reference these methods directly results in unpredictable behavior.

gdl-custom-block-evaluator

Synopsis

gdl-custom-block-evaluator
(*Block*: gdl-custom-block, *Mode*: symbol, *Client*: ui-client-item)

Argument	Description
Block	A user-defined subclass of the permissible custom block classes.
Mode	Either the symbol SYSTEM or MANUAL.
Client	The client for this call, or the object gfr-default-window.

Description

When you create a new class of block, you define its behavior by creating a block evaluator. The evaluator should take as its arguments the class of block for which it is defined, as well as a symbol and a *Client* item. The block evaluator is not a method, but is a procedure customized for a particular class. To associate an evaluator with a class, you also must put the procedure name of your evaluator in the evaluator attribute of the custom class. In this way it serves as a "method" on the class.

If you create your custom class using the custom block wizard, GDA creates an evaluator procedure for you. Code is automatically generated to process the input paths and assign their values to local variables. Generated code also passes the values from local variables onto the output paths. You must write only the code that does the computation and assign it to the proper variables.

If you do not use the wizard, you must create your own evaluator procedure following the format specified here.

The evaluator is called by GDA each time a block is evaluated. You should not call the evaluator directly. Instead, use the API procedure gdl-evaluate-block. The argument indicates whether the thread of the evaluation is launched manually or as part of the normal system evaluation. If the thread is launched manually, the *Client* argument is the client for the user that launches the thread (perhaps by a manual evaluation of a block). Using the *Client* argument, you could launch communications directly on the client for the user. If the mode is SYSTEM, generally you are given the gfr-default-window as the *Client* argument. Except for routing of UI, there is typically no difference between the two modes of evaluation.

gdl-object::gdl-clear-error

Synopsis

gdl-object::gdl-clear-error
(*Obj*: gdl-object)

Argument	Description
Obj	Any block or gdl-defined object.

Description

The gdl-clear-error method clears the error attribute and associated error queue entries and changes the icon background color back to the idle color. A customized error method is only necessary if your evaluator sets an attribute or icon region before signaling an error and you want that setting cleared when the user clears the block error.

Be sure to use call next method in any extensions of this method. Do not call this method directly in any of your code. You should instead use the API, "gdl-clear-block-error" on page 11.

gdl-object::gdl-configure

The gdl-configure method is used to implement the configuration dialogs. It is used to create any additional relations, initialization, or structures that are required when a block is reconfigured. It is not necessary to actually conclude the values into the attributes of your custom block; that is accomplished via call next method. Generally, you would use call next method first, so that the attributes of your block are up to date when the reconfiguration takes place.

Synopsis

gdl-object::gdl-configure
(*CustomBlock*: class gdl-object, *Attributes*: sequence,
Client: class ui-client-item)

Argument	Description
CustomBlock	The Custom Block subclass that you are extending.
Attributes	The sequence of structures describing the attributes to be configured.
Client	The client for this call.

Description

Attributes is a sequence of structures. Each structure represents an attribute that has changed. The structure has attributes attribute-name, the name of the attribute on *CustomBlock*, and attribute-value, the value to be concluded. If gdl-configure is called with an empty *Attributes* sequence, that implies that any or all of the attributes on *CustomBlock* may have been changed previous to the call, so all reconfiguration of the block is to take place.

You can call gdl-configure directly.

Example

In this example, we assume a new class of custom block has been defined, Sample Block with Dual Attribute. It has a public attribute, which enables the user to configure the block with the symbols YES and NO. That value is mirrored by a private attribute, which stores the same information as a truth value. The configuration method ensures that the two attributes remain synchronized when the block is reconfigured.

```
gdl-configure (CustomBlock: class sample-block-with-dual-attribute,
    Attributes: sequence, Client: class ui-client-item)
YesOrNo: symbol;
AttributeStructure: structure;
AttributeFound: truth-value = false;
begin
    call next method;
    if Attributes = sequence () then YesOrNo = the sample-value-as-a-symbol of
        CustomBlock
{New value not passed as an argument, must be obtained directly from the block}
    else begin
        for AttributeStructure = each structure in Attributes do
            if the attribute-name of AttributeStructure = the symbol SAMPLE-
                VALUE-AS-A-SYMBOL then begin
                YesOrNo = the attribute-value of AttributeStructure;
                AttributeFound = true;
            end;
            exit if AttributeFound;
        end;
        if not (AttributeFound) then return; {Attribute was not changed; do nothing}
    end;
    if YesOrNo = the symbol YES then conclude that the _sample-value-as-a-truth
        of CustomBlock = true else conclude that the _sample-value-as-a-truth of
        CustomBlock = false;
end
```

gdl-object::gdl-get-configuration

The gdl-get-configuration method is the converse of gdl-configure. It returns a sequence of attributes that gives all the public attributes for a block, along with their current values. The default implementation uses "gdl-dialog-intermediary" on page 77 associated with your custom block to obtain the public attributes and extract their values. It is only necessary to customize gdl-get-configuration to provide additional information about the configuration of your block. It is not necessary to extend the gdl-get-configuration method even if you have extended gdl-configure. You should put the call next method statement first, and add your additional information to the returned sequence.

Synopsis

gdl-object::gdl-get-configuration
(*CustomBlock*: class gdl-object, *Client*: class ui-client-item)
–> (*sequence*)

Argument	Description
CustomBlock	The custom block subclass you are extending.
Client	The client for this call.

Return Value	Description
sequence	A sequence of structures that describes the current state of the block.

Description

The returned sequence is explained in the description of "gdl-object::gdl-configure" on page 111, a sequence of structures, one structure for each attribute, containing the name and current value of that attribute.

You can call gdl-get-configuration directly. See "gdl-get-configuration" on page 27.

The default implementation of gdl-get-configuration does not return an attribute structure for an attribute that is not a simple attribute. If you have defined an attribute on your *CustomBlock* subclass that contains a subobject (including an array or a g2-variable) and you want the gdl-get-configuration API to return a structure associated with that attribute, you must extend gdl-get-configuration to do it. This is true even if the attribute in question is defined within the proper gdl-dialog-intermediary.

gdl-object::gdl-initialize

Synopsis

gdl-object::gdl-initialize
(*Obj*: gdl-object, *Client*: ui-client-item)

Argument	Description
Obj	Any block or other gdl-defined object.
Client	The client for this call.

Description

The gdl-initialize method creates the structures and relationship necessary for the proper operation of an object. It is called on all gdl-objects when G2 is started. Be sure to use call next method in any extensions of this method.

Do not call this method directly in any of your code. GDA performs the same functions when it starts up.

gdl-object::gdl-reset

Synopsis

gdl-object::gdl-reset
(*Obj*: gdl-object, *Client*: ui-client-item)

Argument	Description
Obj	Any block or other gdl-defined object.
Client	The client for this call.

Description

The gdl-reset method is used to put a gdl-object into a known reset state. It is called on all gdl-objects when G2 is started. Resetting a block calls the gdl-clear-error method, so it is not necessary to duplicate error clearing functions in the reset method. When writing a customized gdl-reset method, you must put the call next method statement first.

Do not call this method directly in any of your code. You should instead use "gdl-reset-item" on page 48.

3

Queues API

Provides a reference for the Queues API.

Introduction

This chapter provides an API reference for programming queues. The API procedures, methods, and functions are organized by the types of objects to which they apply.

This chapter is organized as follows. The chapter describes these queue objects in this order:

- Queues
- Queue Entries
- Queue Entry Sources
- Filters, Subscription, and Logging
- View Managers

- Queue Views
- Buttons

For each type of object:

- The first section contains object definitions, which list the public attributes of that object, as well as the extensible methods and applicable public relations. If you create your own subclass of a given class, the methods listed are those you can use to modify the behavior from that of the parent class. If a method does not appear on the list of extensible methods, then it should not be extended for that class. This is true even if that method is documented as public elsewhere in this document.
- The second section contains all the public procedures for that method. These are the procedures that can be called when extending or modifying the behavior of the queue system.

After these objects are described, the chapter includes sections that describe extensible methods, queue functions, relations, and additional procedures related to these objects.

Queue Classes

The built-in queues are instances of two classes:

This built-in queue..	Is an instance of this class..
Alarm queue	gda-alarm-queue
Explanation queue	gqm-queue
Error queue	gqm-queue
Message queue	gqm-queue

Each type of queue has a setting object, which specifies the global parameters for that type of queue. These setting classes are named *yyy*-*xxx*-queue-setting, where *xxx* is alarm, explanation, error or message, as appropriate. *yyy* references the module that contains the queue definition. For the alarm and explanation queue, this is module gda. For the error and message queue, this is module gqm. Finally, there is a gqm-general setting, which applies over all four queue classes.

Setting objects store the data for the built-in queues, but in the top level module. The classes of these setting objects are named gdabasic-named-queue-setting-*zzz*, where *zzz* is the name of the built-in queue.

gda-alarm-queue

This class extends gqm-queue to add attributes enabling the queue to hold and manipulate entries of the class gda-alarm-entry. The alarm-queue object in GDA is an instance of a gda-alarm-queue.

Attributes of alarm-queue should not be edited directly. Instead, edit the class of settings "gdabasic-named-queue-setting-alarm-queue" on page 144. For instances other than alarm-queue, you can edit attributes directly.

Class Inheritance Path

gda-alarm-queue, gqm-queue, gqs-queue, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
gda-autogenerate-explanation	Generates an explanation at the time of alarm occurrence, which is accessible using the subview.
Allowable values:	Any truth-value
Default value:	false
gda-recurring-entry-class	A symbol referring to a subclass of gda-recurring-alarm-entry, including the default value of gda-recurring-alarm-entry.
Allowable values:	Any symbol
Default value:	GDA-RECURRING-ALARM-ENTRY
gqm-entry-class	The symbol gda-alarm-entry or a symbol naming a subclass of gda-alarm-entry. See "gqm-queue" on page 170 for details.
Allowable values:	Inherited
Default value:	GDA-ALARM-ENTRY

Attribute	Description
gqm-default-priority	See "gqm-queue" on page 170.
Allowable values:	Inherited
Default value:	0
gqm-entry-limit	See "gqm-queue" on page 170.
Allowable values:	Inherited
Default value:	100
gqm-entry-limit-trim-criterion	See "gqm-queue" on page 170.
Allowable values:	Inherited
Default value:	SEVERITY
gqm-display-messages	See "gqm-queue" on page 170.
Allowable values:	Inherited
Default value:	true
gqm-beep-for-new-entry	See "gqm-queue" on page 170.
Allowable values:	Inherited
Default value:	false
gqm-confirm-deletions	See "gqm-queue" on page 170.
Allowable values:	Inherited

Attribute	Description
Default value:	false
gqm-entry-lifetime	See "gqm-queue" on page 170.
Allowable values:	Inherited
Default value:	0.0
gqs-initially-monitor-for-deletion-events	This attribute typically is not used.
Allowable values:	Inherited
Default value:	false
gqs-initially-monitor-for-attribute-change-events	See "gqm-queue" on page 170.
Allowable values:	Inherited
Default value:	false
gqs-item-deletion-callback	See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-attribute-update-callback	See "gqs-queue" on page 176.
Allowable values:	Inherited

Attribute	Description
Default value:	UNSPECIFIED
gqs-item-addition-callback	See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-item-removal-callback	See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-view-template-or-access-table	See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	GDA-DEFAULT-ALARM-QUEUE-VIEW
gqs-logging-manager	This attribute should name an instance of a gda-alarm-logging-manager. See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	GDA-DEFAULT-ALARM-LOG-MANAGER
gqs-update-latency	See "gqm-queue" on page 170.
Allowable values:	Inherited
Default value:	1.0

gda-autogenerate-explanation

Generates an explanation at the time of alarm occurrence, which is accessible using the subview.

Allowable values: Any truth-value

Default value: false

gda-recurring-entry-class

A symbol referring to a subclass of gda-recurring-alarm-entry, including the default value of gda-recurring-alarm-entry.

Allowable values: Any symbol

Default value: GDA-RECURRING-ALARM-ENTRY

gqm-entry-class

The symbol gda-alarm-entry or a symbol naming a subclass of gda-alarm-entry. See "gqm-queue" on page 170 for details.

Allowable values: Inherited

Default value: GDA-ALARM-ENTRY

gqm-default-priority

See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: 0

gqm-entry-limit

See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: 100

gqm-entry-limit-trim-criterion

See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: SEVERITY

gqm-display-messages See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: true

gqm-beep-for-new-entry See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: false

gqm-confirm-deletions See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: false

gqm-entry-lifetime See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: 0.0

gqs-initially-monitor-for-deletion-events This attribute typically is not used.

Allowable values: Inherited

Default value: false

gqs-initially-monitor-for-attribute-change-events See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: false

gqs-item-deletion-callback See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: UNSPECIFIED

gqs-attribute-update-callback See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: UNSPECIFIED

gqs-item-addition-callback See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: UNSPECIFIED

gqs-item-removal-callback See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: UNSPECIFIED

gqs-view-template-or-access-table See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: GDA-DEFAULT-ALARM-QUEUE-VIEW

gqs-logging-manager This attribute should name an instance of a gda-alarm-logging-manager. See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: GDA-DEFAULT-ALARM-LOG-MANAGER

gqs-update-latency See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: 1.0

gda-autogenerate-explanation Generates an explanation at the time of alarm occurrence, which is accessible using the subview.

Allowable values: Any truth-value

Default value: false

gda-recurring-entry-class A symbol referring to a subclass of gda-recurring-alarm-entry, including the default value of gda-recurring-alarm-entry.

Allowable values: Any symbol

Default value: GDA-RECURRING-ALARM-ENTRY

gqm-entry-class The symbol gda-alarm-entry or a symbol naming a subclass of gda-alarm-entry. See "gqm-queue" on page 170 for details.

Allowable values: Inherited

Default value: GDA-ALARM-ENTRY

gqm-default-priority See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: 0

gqm-entry-limit See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: 100

gqm-entry-limit-trim-criterion See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: SEVERITY

gqm-display-messages See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: true

gqm-beep-for-new-entry See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: false

gqm-confirm-deletions See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: false

gqm-entry-lifetime See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: 0.0

gqs-initially-monitor-for-deletion-events This attribute typically is not used.

Allowable values: Inherited

Default value: false

gqs-initially-monitor-for-attribute-change-events See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: false

gqs-item-deletion-callback See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: UNSPECIFIED

gqs-attribute-update-callback See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: UNSPECIFIED

gqs-item-addition-callback See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: UNSPECIFIED

gqs-item-removal-callback See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: UNSPECIFIED

gqs-view-template-or-access-table See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: GDA-DEFAULT-ALARM-QUEUE-VIEW

gqs-logging-manager This attribute should name an instance of a gda-alarm-logging-manager. See "gqs-queue" on page 176.

Allowable values: Inherited

Default value: GDA-DEFAULT-ALARM-LOG-MANAGER

gqs-update-latency See "gqm-queue" on page 170.

Allowable values: Inherited

Default value: 1.0

Extensible Methods

"gda-alarm-queue::gqs-remove-items" on page 352

"gqm-queue::gqs-receive-items" on page 365

"gqm-queue::gqs-remove-items" on page 366

"gqs-queue::gqs-clear-queue" on page 369

"gqs-queue::gqs-receive-items" on page 370

gda-alarm-queue-setting

This class holds preferences applicable across all alarm queues in the application. This is the programmatic equivalent of configuring the alarm queue through the Preferences > Queues > Alarm main menu choice. See "gdl-get-active-setting" on page 26 for additional information.

Class Inheritance Path

gda-alarm-queue-setting, gqm-queue-setting, gfr-module-setting, object, item

Attributes

Attribute	Description
gda-alarm-log-formatter	Names a procedure of the format as described for "gda-format-alarm-log-text" on page 261.
Allowable values:	Any symbol
Default value:	GDA-FORMAT-ALARM-LOG-TEXT
gda-reuse-entry	Whether or not to use an existing entry when an alarm source re-enters alarm.
Allowable values:	Any truth-value
Default value:	true
gda-history-limit	The number of alarm transitions to store in history for a single entry.
Allowable values:	Any integer
Default value:	50

Attribute	Description
gqm-queue-category	See "gqm-queue-setting" on page 175.
Allowable values:	The symbol ALARM-QUEUE
Default value:	ALARM-QUEUE
gqm-standard-queue	See "gqm-queue-setting" on page 175.
Allowable values:	Inherited
Default value:	ALARM-QUEUE

gda-explanation-queue-setting

This class holds preferences applicable across all explanation queues in the application. This is the programmatic equivalent of configuring the explanation queue through the Preferences > Queues > Explanation main menu choice. See "gdl-get-active-setting" on page 26 for additional information.

Class Inheritance Path

gda-explanation-queue-setting, gqm-queue-setting, gfr-module-setting, object, item

Attributes

Attribute	Description
gqm-queue-category	See "gqm-queue-setting" on page 175.
Allowable values:	the symbol EXPLANATION-QUEUE
Default value:	EXPLANATION-QUEUE
gqm-standard-queue	See "gqm-queue-setting" on page 175.
Allowable values:	Inherited
Default value:	EXPLANATION-QUEUE

gdabasic-named-queue-setting

This setting object mirrors the attributes of the built-in queue objects and the default alarm logging managers. This enables a user of the queue system to store non-default configurations of the built-in queues and built-in logging managers in a higher level module. On startup, the attribute values from the applicable setting instance (see gfr-get-active-setting in the GFR documentation) are installed in the appropriate queue.

This setting class is not instantiated. Instead, subclasses of this setting are used. The subclasses are gdabasic-named-queue-setting-alarm-queue, gdabasic-named-queue-setting-explanation-queue, gdabasic-named-queue-setting-error-queue, and gdabasic-named-queue-setting-message-queue. With the exception of the alarm queue, all setting classes have the same attributes.

When accessing these settings to be written to programmatically, be sure to use the API "gdl-get-active-setting" on page 26. You may also need to call the API gfr-propagate-module-setting-information (see the GFR documentation) to propagate values from the setting to the queue.

Class Inheritance Path

gdabasic-named-queue-setting, gfr-module-setting, object, item

Attributes

Attribute	Description
gqm-entry-class	See "gqm-queue" on page 170.
Allowable values:	Any symbol
Default value:	GQM-ENTRY
gqm-entry-limit	See "gqm-queue" on page 170.
Allowable values:	Any integer
Default value:	100
gqm-display-messages	See "gqm-queue" on page 170.
Allowable values:	Any truth-value

Attribute	Description
Default value:	true
gqm-beep-for-new-entry	See "gqm-queue" on page 170.
Allowable values:	Any truth-value
Default value:	false
gqm-confirm-deletions	See "gqm-queue" on page 170.
Allowable values:	Any truth-value
Default value:	false
gqs-view-template-or-access-table	See "gqm-queue" on page 170.
Allowable values:	Any symbol
Default value:	GDABASIC-DEFAULT-MESSAGE-QUEUE-VIEW
gqs-logging-manager	See "gqm-queue" on page 170.
Allowable values:	Any symbol
Default value:	GDABASIC-DEFAULT-MESSAGE-LOG-MANAGER
gqs-update-latency	See "gqm-queue" on page 170.
Allowable values:	Any float
Default value:	1.0

Attribute	Description
gqm-default-priority	See "gqm-queue" on page 170.
Allowable valuves:	Any integer
Default value:	0
gqm-entry-lifetime	See "gqm-queue" on page 170.
Allowable values:	Any float
Default value:	0.0
gqs-item-deletion-callback	See "gqm-queue" on page 170.
Allowable valuves:	Any symbol
Default value:	UNSPECIFIED
gqs-attribute-update-callback	See "gqm-queue" on page 170.
Allowable values:	Any symbol
Default value:	UNSPECIFIED
gqs-item-addition-callback	See "gqm-queue" on page 170.
Allowable values:	Any symbol
Default value:	UNSPECIFIED
gqs-item-removal-callback	See "gqm-queue" on page 170.
Allowable values:	Any symbol

Attribute	Description
Default value:	UNSPECIFIED
glf-logging-enabled	See "gqm-logging-manager" on page 251.
Allowable values:	Any truth-value
Default value:	false
glf-log-directory	See "gqm-logging-manager" on page 251.
Allowable values:	Any text
Default value:	""
glf-log-file-name-template	See "gqm-logging-manager" on page 251.
Allowable values:	Any text
Default value:	"log_*.txt"
glf-log-file-name-generator	See "gqm-logging-manager" on page 251.
Allowable values:	Any symbol
Default value:	GLF-DEFAULT-FILE-NAME-GENERATOR
glf-file-header-writer	See "gqm-logging-manager" on page 251.
Allowable values:	Any symbol
Default value:	GLF-DEFAULT-LOG-FILE-HEADER-WRITER
glf-time-interval-to-open-new-log-file	See "gqm-logging-manager" on page 251.
Allowable values:	Any integer

Attribute	Description
Default value:	86400
glf-maximum-file-size-in-bytes	See "gqm-logging-manager" on page 251.
Allowable values:	Any integer
Default value:	100000
glf-log-file-scheduler	See "gqm-logging-manager" on page 251.
Allowable values:	Any symbol
Default value:	GLF-DEFAULT-LOG-FILE-SCHEDULER
glf-automatically-delete-empty-log-files	See "gqm-logging-manager" on page 251.
Allowable values:	Any truth-value
Default value:	true
gqm-time-format	See "gqm-logging-manager" on page 251.
Allowable values:	Any symbol
Default value:	DATETIME
gqm-log-comments	See "gqm-logging-manager" on page 251.
Allowable values:	Any truth-value
Default value:	false
gdabasic-dialog-to-launch	See "gqm-logging-manager" on page 251.

Attribute	Description
Allowable values:	Any text
Default value:	"gdi-queue"

gdabasic-named-queue-setting-alarm-queue

This subclass adds attributes that exist only on the gda-alarm-queue class or on the gda-alarm-logging-manager class.

Class Inheritance Path

gdabasic-named-queue-setting-alarm-queue, gdabasic-named-queue-setting, gfr-module-setting, object, item

Attributes

Attribute	Description
gda-autogenerate-explanation	See "gda-alarm-queue" on page 124.
Allowable values:	Any truth-value
Default value:	false
gda-recurring-entry-class	See "gda-alarm-queue" on page 124.
Allowable values:	Any symbol
Default value:	GDA-RECURRING-ALARM-ENTRY
gda-log-events-incrementally	See "gda-alarm-logging-manager" on page 245.
Allowable values:	Any truth-value
Default value:	false
gda-log-explanations	See "gda-alarm-logging-manager" on page 245.
Allowable values:	Any truth-value
Default value:	false

Attribute	Description
gda-log-advice	See "gda-alarm-logging-manager" on page 245.
Allowable values:	Any truth-value
Default value:	false
gqm-entry-class	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GDA-ALARM-ENTRY
gqm-entry-limit	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	100
gqm-display-messages	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	true
gqm-beep-for-new-entry	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false
gqm-confirm-deletions	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited

Attribute	Description
Default value:	false
gqs-view-template-or-access-table	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GDA-DEFAULT-ALARM-QUEUE-VIEW
gqs-logging-manager	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GDA-DEFAULT-ALARM-LOG-MANAGER
gqs-update-latency	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	1.0
gqm-default-priority	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	0
gqm-entry-lifetime	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	0.0

Attribute	Description
gqs-item-deletion-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-attribute-update-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-item-addition-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-item-removal-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
glf-logging-enabled	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false
glf-log-directory	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited

Attribute	Description
Default value:	""
glf-log-file-name-template	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	"loga_*.txt"
glf-log-file-name-generator	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-FILE-NAME-GENERATOR
glf-file-header-writer	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-LOG-FILE-HEADER-WRITER
glf-time-interval-to-open-new-log-file	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	86400
glf-maximum-file-size-in-bytes	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited

Attribute	Description
Default value:	100000
glf-log-file-scheduler	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-LOG-FILE-SCHEDULER
glf-automatically-delete-empty-log-files	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	true
gqm-time-format	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	DATETIME
gqm-log-comments	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false
gdabasic-dialog-to-launch	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	"gda-queue"

gdabasic-named-queue-setting-error-queue

Class Inheritance Path

gdabasic-named-queue-setting-error-queue, gdabasic-named-queue-setting, gfr-module-setting, object, item

Attributes

Attribute	Description
gqm-entry-class	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GQM-ERROR-ENTRY
gqm-entry-limit	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	100
gqm-display-messages	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	true
gqm-beep-for-new-entry	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false

Attribute	Description
gqm-confirm-deletions	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false
gqs-view-template-or-access-table	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GDABASIC-DEFAULT-ERROR-QUEUE-VIEW
gqs-logging-manager	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GDABASIC-DEFAULT-ERROR-LOG-MANAGER
gqs-update-latency	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	1.0
gqm-default-priority	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	0
gqm-entry-lifetime	See "gdabasic-named-queue-setting" on page 138.

Attribute	Description
Allowable values:	Inherited
Default value:	0.0
gqs-item-deletion-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-attribute-update-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-item-addition-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-item-removal-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
glf-logging-enabled	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited

Attribute	Description
Default value:	false
glf-log-directory	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	""
glf-log-file-name-template	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	"loge_*.txt"
glf-log-file-name-generator	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-FILE-NAME-GENERATOR
glf-file-header-writer	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-LOG-FILE-HEADER-WRITER
glf-time-interval-to-open-new-log-file	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited

Attribute	Description
Default value:	86400
glf-maximum-file-size-in-bytes	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	100000
glf-log-file-scheduler	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-LOG-FILE-SCHEDULER
glf-automatically-delete-empty-log-files	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	true
gqm-time-format	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	DATETIME
gqm-log-comments	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false

Attribute	Description
gdabasic-dialog-to-launch	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	"gdabasic-queue"

gdabasic-named-queue-setting-explanation-queue

Class Inheritance Path

gdabasic-named-queue-setting-explanation-queue, gdabasic-named-queue-setting, gfr-module-setting, object, item

Attributes

Attribute	Description
gqm-entry-class	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GDA-EXPLANATION-ENTRY
gqm-entry-limit	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	100
gqm-beep-for-new-entry	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false
gqm-confirm-deletions	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false

Attribute	Description
gqs-view-template-or-access-table	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GDA-DEFAULT-EXPLANATION-QUEUE-VIEW
gqs-logging-manager	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GDA-DEFAULT-EXPLANATION-LOG-MANAGER
gqs-update-latency	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	1.0
gqm-default-priority	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	0
gqm-entry-lifetime	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	0.0
gqs-item-deletion-callback	See "gdabasic-named-queue-setting" on page 138.

Attribute	Description
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-attribute-update-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-item-addition-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-item-removal-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
glf-logging-enabled	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false
glf-log-directory	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited

Attribute	Description
Default value:	""
glf-log-file-name-template	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	"logx_*.txt"
glf-log-file-name-generator	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-FILE-NAME-GENERATOR
glf-file-header-writer	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-LOG-FILE-HEADER-WRITER
glf-time-interval-to-open-new-log-file	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	86400
glf-maximum-file-size-in-bytes	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited

Attribute	Description
Default value:	100000
glf-log-file-scheduler	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-LOG-FILE-SCHEDULER
glf-automatically-delete-empty-log-files	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	true
gqm-time-format	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	DATETIME
gqm-log-comments	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false
gdabasic-dialog-to-launch	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	"gdi-queue"

gdabasic-named-queue-setting-message-queue

Class Inheritance Path

gdabasic-named-queue-setting-message-queue, gdabasic-named-queue-setting, gfr-module-setting, object, item

Attributes

Attribute	Description
gqm-entry-class	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GQM-ENTRY
gqm-entry-limit	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	100
gqm-display-messages	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	true

Attribute	Description
gqm-beep-for-new-entry	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false
gqm-confirm-deletions	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false
gqs-view-template-or-access-table	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GDABASIC-DEFAULT-MESSAGE-QUEUE-VIEW
gqs-logging-manager	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GDABASIC-DEFAULT-MESSAGE-LOG-MANAGER
gqs-update-latency	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	1.0
gqm-default-priority	See "gdabasic-named-queue-setting" on page 138.

Attribute	Description
Allowable values:	Inherited
Default value:	0
gqm-entry-lifetime	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	0.0
gqs-item-deletion-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-attribute-update-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-item-addition-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-item-removal-callback	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited

Attribute	Description
Default value:	UNSPECIFIED
glf-logging-enabled	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false
glf-log-directory	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	""
glf-log-file-name-template	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	"logm_*.txt"
glf-log-file-name-generator	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-FILE-NAME-GENERATOR
glf-file-header-writer	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited

Attribute	Description
Default value:	GLF-DEFAULT-LOG-FILE-HEADER-WRITER
glf-time-interval-to-open-new-log-file	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	86400
glf-maximum-file-size-in-bytes	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	100000
glf-log-file-scheduler	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-LOG-FILE-SCHEDULER
glf-automatically-delete-empty-log-files	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	true
gqm-time-format	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	DATETIME

Attribute	Description
gqm-log-comments	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	false
gdabasic-dialog-to-launch	See "gdabasic-named-queue-setting" on page 138.
Allowable values:	Inherited
Default value:	"gdabasic-queue"

gqm-error-queue-setting

To the functionality contained in gqm-queue-setting, this adds a global preference that applies only to error queues. For more details, see the documentation of the error queue and error entries. Use "gdl-get-active-setting" on page 26 to access this module setting for writing.

Class Inheritance Path

gqm-error-queue-setting, gqm-queue-setting, gfr-module-setting, object, item

Attributes

Attribute	Description
gqm-show-tracebacks	Whether or not to display the G2-generated traceback (if available) within the message of the error entry.
Allowable values:	Any truth-value
Default value:	true
gqm-queue-category	See "gqm-queue-setting" on page 175
Allowable values:	The symbol ERROR-QUEUE
Default value:	ERROR-QUEUE
gqm-standard-queue	See "gqm-queue-setting" on page 175
Allowable values:	Inherited
Default value:	ERROR-QUEUE

gqm-general-setting

This module setting provides programmatic access to the preferences configured through the main menu, Preferences > Queues > General Settings. Use "gdl-get-active-setting" on page 26 when getting an instance of this setting for writing.

The following substitution strings are the allowable values for the templates referred to in the table below.

Substitution strings

<y2> - 2-digit year

<y4> - 4-digit year

<d> - day

<d2> - 2-digit day

<df> - full day name

<da> - abbreviated day name

<dor> - ordinal day

<m> - month

<m2> - 2-digit month

<mf> - full month name

<ma> - abbreviated month name

<apm> - A.M. or P.M.

<h12> - hour (12 hour)

<h24> - hour (24 hour)

<min> - minute

<sec> - second

Class Inheritance Path

gqm-general-setting, gfr-module-setting, object, item

Attributes

Attribute	Description
gqm-date-time-format	Provides a template for writing out the time and the date.
Allowable values:	Any text
Default value:	""
gqm-date-format	Provides a template for writing out the date.
Allowable values:	Any text
Default value:	""
gqm-time-format	Provides a template for writing out the time.
Allowable values:	Any text
Default value:	""
gqm-file-time-format	Provides a template for writing out the time and the date.
Allowable values:	Any text
Default value:	""
gqm-ordination-procedure	A symbol naming a procedure of the format specified in "gqm-default-ordination" on page 397.
Allowable values:	Any symbol
Default value:	GQM-DEFAULT-ORDINATION

gqm-queue

The gqm-queue class is the basic class for all the queues within GDA. The error queue, explanation queue, and message queue are instances of this class.

You should not edit attributes of the built-in queues (the queues supplied with GDA), either manually or programmatically, directly. For manual edits, access the configuration dialog for the queue. For programmatic edits, edit the appropriate subclass of "gdabasic-named-queue-setting" on page 138 instead. For instances other than the built-in queues, you can edit attributes on the queue directly.

For attributes described for gqm-queue and the queue hierarchy, the effects of changing the attributes (i.e. configuring the queue) are described more thoroughly in the *GDA User's Guide*.

You can subclass and extend this class and the subclass gda-alarm-queue. You can specialize this class by configuring the callback attributes. It is important to know that queue behavior can be specialized by using other classes. The gqm-queue class should only be subclassed if you are adding attributes or extending the gqm-queue::gqs-clear-queue method. Further, only place customizations on the queue (either in the methods or the callbacks) if the custom actions are on the data server side. See the class "gqs-view-manager" on page 287 for a way to customize the queue actions using a subscribed view. Finally, if the customizations are associated with additional entry attributes that need to be handled, it might be most appropriate to subclass "gqm-entry" on page 213.

Class Inheritance Path

gqm-queue, gqs-queue, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
gqm-entry-class	The symbol gqm-entry or a symbol naming any subclass of gqm-entry. This attribute specifies what class of entry to create when a new entry is posted to the queue using gqm-post-entry.
Allowable values:	Any symbol
Default value:	GQM-ENTRY

Attribute	Description
gqm-default-priority	This attribute is not used by GDA built-in queues. The default priority is assigned to the gqm-priority of the gqm-entry created by the queue. This attribute is available for application developer use.
Allowable values:	Any integer
Default value:	0
gqm-entry-limit	This integer specifies the number of entries that can be maintained in the queue. If additional entries arrive, the oldest is discarded.
Allowable values:	Any integer
Default value:	100
gqm-display-messages	When this attribute is true, receipt of a new entry on the queue launches a view on all ui-client-items that are connected.
Allowable values:	Any truth-value
Default value:	true
gqm-beep-for-new-entry	When this attribute is true, showing a new view is accompanied by a beep.
Allowable values:	Any truth-value
Default value:	false
gqm-confirm-deletions	Whether or not to launch a dialog that confirms deletion of the selected entries.
Allowable values:	Any truth-value
Default value:	false

Attribute	Description
gqm-entry-lifetime	This attribute is only used by the explanation queue and the alarm queue. When a new explanation or alarm entry is created, it is set to expire in the number of seconds specified in this attribute. If the value of this attribute is 0, the entries never expire.
Allowable values:	Any float
Default value:	0.0
gqs-initially-monitor-for-deletion-events	See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	false
gqs-initially-monitor-for-attribute-change-events	Typically this attribute is not used. See "gqs-queue" on page 176 for more details.
Allowable values:	Inherited
Default value:	false
gqs-item-deletion-callback	See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-attribute-update-callback	See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	UNSPECIFIED

Attribute	Description
gqs-item-addition-callback	See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-item-removal-callback	See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-view-template-or-access-table	See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-logging-manager	This attribute should name an instance of a gqm-logging-manager. See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqs-update-latency	See "gqs-queue" on page 176.
Allowable values:	Inherited
Default value:	1.0

Extensible Methods

"gqm-queue::gqs-receive-items" on page 365

"gqm-queue::gqs-remove-items" on page 366

"gqs-queue::gqs-clear-queue" on page 369

"gqs-queue::gqs-receive-items" on page 370

Relations

gqmv-being-worked-on

gqm-queue-setting

This setting contains the name of the queue launched from the main menu.

Class Inheritance Path

gqm-queue-setting, gfr-module-setting, object, item

Attributes

Attribute	Description
gqm-queue-category	This attribute provides a symbol lookup so the menu system can determine with which of the four queues this setting is associated. This attribute is read-only.
Allowable values:	The symbol MESSAGE-QUEUE
Default value:	MESSAGE-QUEUE
gqm-standard-queue	The queue named in this setting is the queue launched from the main menu. This attribute is read/write.
Allowable values:	Any symbol that names a queue of the correct class.
Default value:	MESSAGE-QUEUE

gqs-queue

This is the basic foundation class for all of the queues. The object stores an ordered list of items. Each queue can support a number of views, but does not itself have a visual representation, aside from its icon. This class should not be extended by the GDA user. It is included in the documentation for reference only.

Class Inheritance Path

gqs-queue, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
gqs-initially-monitor-for-deletion-events	Controls whether the queue monitors for the deletion of the items it contains, when G2 starts. This is an attribute that is applicable to a GQS queue, which can contain items of any class. This does not apply to GQM queues, which contain only gqm-entries.
Allowable values:	Should be kept false for any GDA or NOL application.
Default value:	false
gqs-initially-monitor-for-attribute-change-events	Controls whether the queue monitors the attributes of the items it contains, when G2 starts. For any predefined GDA entry attributes, this should typically be false. If your entry class contains additional attributes that need to be monitored, this attribute may come into play. To turn monitoring on or off after G2 has started, use "gqs-activate-attribute-monitoring" on page 186 or "gqs-deactivate-attribute-monitoring" on page 189.
Allowable values:	Any truth value
Default value:	false
gqs-item-deletion-callback	Because removal in GDA implies deletion (when the gqm-entry in question no longer resides on any queues), this callback would generally never be used. See "gqs-remove-items" on page 198 for a description of this behavior.

Attribute	Description
Allowable values:	The name of the procedure to be called when an item contained by the queue is deleted, or the symbol UNSPECIFIED.
Default value:	UNSPECIFIED
gqs-attribute-update-callback	The name of the procedure to be called when an attribute of an item contained by the queue receives a value. See also "gqs-view-manager" on page 287 and "gqm-entry" on page 213 for alternate ways to customize the behavior of the queues. Your callback must take three arguments: Queue: class gqs-queue, MonitoredItem: class item, AttributeName: symbol. Also, the attribute must either be monitored (see "gqs-add-monitored-attributes" on page 187) or be one of the built-in attributes described under "gda-alarm-entry" on page 204.
Allowable values:	The name of the procedure to be called when an item contained by the queue is updated, or the symbol UNSPECIFIED.
Default value:	UNSPECIFIED
gqs-item-addition-callback	The name of a procedure to be called when items are added to the queue. Use this procedure if you want to take an action, on a per-queue basis, as new entries are added to the queue. To simply be notified of entries arriving on the queue, see the description of "gqs-view-manager" on page 287. If you have created your own subclass of gqm-entry, and need to create the data structures that go with it, see "gqm-entry::gqm-entry-constructor" on page 359. Finally, note that callbacks cannot be further extended by your users. If you want to modify what happens upon addition, but want the system to remain open for further extensions, you may want to subclass the queue and extend the method gqm-queue::gqs-update-per-addition. If you do write a callback for the queue, your callback must accept three arguments: Queue: class gqs-queue, ItemsAdded: class item-list, Client: class object. The last argument is the same client object passed to the procedure that added the items to the queue.

Attribute	Description
Allowable values:	The name of the procedure to be called when one or more items are added to the queue, or the symbol UNSPECIFIED.
Default value:	UNSPECIFIED
gqs-item-removal-callback	The name of a procedure to be called when items are removed from the queue. Use this procedure if you want to take some action, on a per-queue basis, before an entry gets removed. To simply be notified of entry removals from the queue, see the description of "gqs-view-manager" on page 287. If you have created your own subclass of gqm-entry, and need to clean up data structures associated with it, see "gqm-entry::gqm-delete" on page 358. Finally, note that callbacks cannot be further extended by your users. If you want to modify what happens upon removal, but want the system to remain open for further extensions, you may want to subclass the queue and extend the method gqm-queue::gqs-update-per-removal. If you do write a callback for the queue, your callback must accept three arguments: Queue: class gqs-queue, ItemsRemoved: class item-list, Client: class object. The last argument is the same client object passed to the procedure that removed the items from the queue.
Allowable values:	The name of the procedure to be called when one or more items contained by the queue are removed, or the symbol UNSPECIFIED.
Default value:	UNSPECIFIED
gqs-view-template-or-access-table	The name of the gqs-queue-access-table that controls what template should be used to create a view of the queue, on the current window, or the name of an item that implements the method gqs-create-view. If this attribute is unspecified, the menu choice launch-view does not appear on the queue. The built-in class gqmv-tabular-view-template implements this procedure.

Attribute	Description
Allowable values:	Any symbol naming either a gqs-queue-access-table, a gqmv-tabular-view-template, or other item as described below.
Default value:	UNSPECIFIED
gqs-logging-manager	The name of an optional logging manager for the queue, which must be an instance of a glf-logging-manager.
Allowable values:	Any symbol
Default value:	UNSPECIFIED
gqs-update-latency	The latency, in seconds, between updates when items are added or removed from the queue. If this interval is too short, the efficiency of the system may suffer, because the queue is not able to treat multiple insertions or removals as a group.
Allowable values:	Any positive float
Default value:	1.0

Extensible Methods

"gqs-queue::gqs-clear-queue" on page 369

"gqs-queue::gqs-receive-items" on page 370

Relations

gqs-view-manager-for-queue

Queue Procedures

This section describes the procedures that support the queue system.

gda-set-auto-explain

Synopsis

gda-set-auto-explain
(*YesNo*: truth-value)

Argument	Description
YesNo	Value to set generate-automatic-explanation on alarm.

Description

Iterates through all subclasses of gdl-alarm-source, setting the attribute generate-automatic-explanation to equal the value *YesNo*. This has the effect of globally setting the automatic explanation for all alarms.

gqm-get-queue-names

Synopsis

gqm-get-queue-names
()
–> *Sequence*

Return Value	Description
Sequence	A sequence of symbols, the queue names.

Description

Returns a sequence of the names of every instance of gqm-queue (or subclass) within the application.

gqm-post-entry

Synopsis

gqm-post-entry
(*Queue*: gqm-queue, *EntryData*: structure, *Client*: object)
–> *gqm-Entry*

Argument	Description
Queue	The queue to which a new entry is to be posted.
EntryData	A structure that contains the data to be used in creating the new entry (see below).
Client	The client or window object.

Return Value	Description
gqm-Entry	A newly created entry of the class named by the gqm-entry-class of *Queue*.

Description

This procedure creates and returns a new entry. The entry is initialized via its gqm-entry-constructor method and then inserted onto *Queue*.

The class of the newly created entry is determined by the gqm-entry-class of *Queue* except if the gqm-source of the *EntryData* is a gdl-recurring-alarm. If this is the case, the gda-alarm-queue-attribute, gda-recurring-entry-class, is used to determine the class of the entry.

See also:

"gda-alarm-entry::gqm-entry-constructor" on page 350

"gda-entry-with-uuid::gqm-entry-constructor" on page 353

"gda-recurring-alarm-entry::gqm-entry-constructor" on page 355

"gqm-entry::gqm-entry-constructor" on page 359

"gqm-error-entry::gqm-entry-constructor" on page 364

EntryData Structure Description

The required attributes in the *EntryData* data structure are determined by the class of entry that is ultimately to be created. All entries require a gqm-text (see the gqm-error attribute for a caveat in the case of error entries). All queues except the message queue (which uses entry class gqm-entry) also use the attribute gqm-source. The remaining attributes apply only to gda-alarm-queue.

Attribute	Type	Description
gqm-text	Text	Contains the text of the entry that is to be displayed on the queue view.
gqm-error	Error	When posting to a queue that generated error entries, an error itself can be included in the entry data as an alternative to gqm-text. If this is the case, the attribute gqm-text must be omitted.
gqm-source	Item	For gqm-entry, this is not required. For other entries, this attribute is to contain the source of the data. In the case of gda-alarm-entry, the source must be of the class gdl-alarm-source. The source is linked to the entry via the relation the-gqm-source-of.
gda-status	Symbol	This applies only to gda-alarm-entry. If gqm-post-entry is called by a GDA object, the status is derived from the belief value. A GDA block uses the Output Uncertainty to convert belief values to status. If logic is discrete, then the belief value must be either 0.0, 0.5, or 1.0. The status value should be assigned according to the belief value with .TRUE accompanying 1.0, UNKNOWN with 0.5, and .FALSE with 0.0.
gda-belief	Float	This applies only to gda-alarm-entry. This is the fuzzy belief value of the alarm, with a value from 0.0 to 1.0. Regardless of whether the logic is discrete or fuzzy, a belief value must be included in the gqm-post-entry call. If the logic is discrete, then pass a value of either 0.0 or 1.0 (or 0.5 if three-way logic is being used, and you want to specify a belief on unknown).

Attribute	Type	Description
gda-collection-time	Float	This applies only to gda-alarm-entry. The attribute contains a collection time (in G2 time, as a float) of the data which caused the alarm.
gda-severity	Integer	This applies only to gda-alarm-entry. This is an integer that corresponds to the severity of the alarm

Applicable methods

"gqm-entry::gqm-entry-constructor" on page 359

gqs-activate-attribute-monitoring

Synopsis

gqs-activate-attribute-monitoring
(*Queue*: gqs-queue, *Client*: object)

Argument	Description
Queue	The target queue.
Client	The client for this call.

Description

This procedure activates attribute monitoring for the current set of monitored attributes. When attribute monitoring is active, the queue sends attribute-change events to its views, so they reflect the current state of the items in the queue. To specify the monitored attributes, see "gqs-add-monitored-attributes" on page 187.

Use of attribute monitoring may have a noticeable effect on performance if the number of items in the queue is large, and it may be possible to achieve the same goal by monitoring attribute changes at the view level, which is much more efficient. Also see "gqs-deactivate-attribute-monitoring" on page 189.

gqs-add-monitored-attributes

Synopsis

gqs-add-monitored-attributes
(*Queue*: gqs-queue, *Attributes*: symbol-list, *Client*: object)

Argument	Description
Queue	The queue that contains items to be monitored.
Attributes	A list of attributes to add to the set of monitored attributes.
Client	The client for this call.

Description

Use this procedure to set or extend the set of attributes monitored in the items contained in *Queue*. See also "gqs-remove-monitored-attributes" on page 199 and "gqs-remove-all-monitored-attributes" on page 197. Adding attributes using this procedure does not start monitoring if it is not already on. To turn on attribute monitoring, use "gqs-activate-attribute-monitoring" on page 186.

gqs-clear-queue

Synopsis

gqs-clear-queue
(*Queue*: gqs-queue, *Client*: object)

Argument	Description
Queue	The queue to be cleared
Client	The client for this call.

Description

See "gqs-queue::gqs-clear-queue" on page 369.

gqs-deactivate-attribute-monitoring

Synopsis

gqs-deactivate-attribute-monitoring
(*Queue*: gqs-queue, *Client*: object)

Argument	Description
Queue	The queue that is the target of this call.
Client	The client for this call.

Description

This procedure turns off attribute monitoring on the *Queue*. See "gqs-activate-attribute-monitoring" on page 186 for more details.

gqs-force-input-buffer-into-queue

Synopsis

gqs-force-input-buffer-into-queue
(*Queue*: gqs-queue, *Client*: object)

Argument	Description
Queue	The queue that may contain items in its input buffer.
Client	The client for this call.

Description

For efficiency, each queue maintains a list of items waiting to be added to the queue. Periodically, with a latency determined by the gqs-update-latency of the queue, the items in this list are added as a group to the queue. Occasionally, you may need to force the items from the input buffer into the queue using this procedure, before performing some operation.

For example, if you programmatically send items to the queue, and them programmatically sort the items, you need to call this procedure to assure that all items sent to the queue are actually incorporated into the queue when you perform the sort.

gqs-get-collected-items

Synopsis

gqs-get-collected-items
(*Queue*: gqs-queue, *ItemList*: item-list, *Client*: object)

Argument	Description
Queue	The queue whose items are to be accessed.
ItemList	The item list that contains the results.
Client	The client for this call.

Description

Adds a queue's collected items to the end of *ItemList*. The list *ItemList* is not cleared before this operation. This procedure enables you to obtain a list of all items in a queue. For a GDA queue, the items in the list are of class gqm-entry.

gqs-get-monitored-attributes

Synopsis

gqs-get-monitored-attributes
(*Queue*: gqs-queue, *Attributes*: symbol-list, *Client*: object)

Argument	Description
Queue	The queue that is the target of this call.
Attributes	The list to contain the results.
Client	The client for this call.

Description

This procedure returns the set of monitored attributes for the given queue. The symbols are appended to *Attributes* without clearing the list first.

gqs-get-queues-containing-item

Synopsis

gqs-get-queues-containing-item
(*Item*: item, *QueueList*: item-list, *Client*: object)

Argument	Description
Item	The item that may be contained in one or more queues.
QueueList	The list that is to contain the results.
Client	The client for this call.

Description

This procedure returns in *QueueList* all the queues that currently contain the given *Item*. *QueueList* is not cleared before the results are inserted.

gqs-launch-view

Synopsis

gqs-launch-view
(*Queue*: gqs-queue, *Client*: object)
–> *gqs-View-Manager*

Argument	Description
Queue	The queue that is being asked to launch a view.
Client	The window or other client object where the view is to be launched.

Return Value	Description
gqs-View-Manager	A newly created view manager.

Description

This procedure is used to create a new view manager associated with the given queue.

- If the gqs-view-template-or-access-table of *Queue* contains the name of a queue access table, gqs-get-view-template is used to determine the proper view template for the given client.
- Otherwise, the gqs-view-template-or-access-table of the *Queue* must name an item that can be used as a view template (any item implementing gqs-create-view). Once the template has been identified, the method gqs-create-view is called to create the view manager, which is returned by this procedure.

gqs-receive-items

Synopsis

gqs-receive-items
(*Queue*: gqm-queue, *Sender*: item-or-value, *Entries*: item-list,
Client: object)

Argument	Description
Queue	The queue receiving items.
Sender	Typically false. Not used by this subclass.
Entries	A list of the new entries to be added to the queue.
Client	The client for this call.

Description

Call this procedure to add a list of items to *Queue*. Note that calling this procedure does not immediately add the items to the queue; instead, it adds them to the queue's input buffer for later insertion. For more information, see "gqs-force-input-buffer-into-queue" on page 190. This procedure does use the *Sender*, which is provided for user overrides of this method, but is not used by default.

GQM extends the GQS behavior by adding three functions.

1. First, if the gqm-queue is set up to be logging entries, the entry is logged by making a call to the method gqm-log-entry on the class of the entry received.
2. Second, the procedure compares the number of items that the queue will have at the end of the receiving process with the limit set for the queue in gqm-entry-limit. If the limit is exceeded, the entries are forced into the queue and the oldest entries thrown out.
3. Finally, if the gqm-messages-display for *Queue* is true, the receipt of new entries launches a new view on any *Client* that does not already have a view. Note that this means if a queue exists but is hidden behind other workspaces, it is not shown.

gqs-receive-single-item

Synopsis

gqs-receive-single-item
(*Receiver*: gqs-queue, *Item*: item, *Client*: object)

Argument	Description
Receiver	The queue to receive the item.
Item	The item to be introduced into the queue.
Client	The client for this call.

Description

Use this procedure to introduce a new item to a queue, without an explicit sender. Note that there is a latency between the receipt of an item by a queue, and the actual incorporation of the item into the queue. See "gqs-force-input-buffer-into-queue" on page 190 for details. If you need to introduce multiple items into a queue at one time, it is more efficient to use "gqs-receive-items" on page 195. See also "gqs-send-single-item" on page 202.

Applicable Methods

"gqm-queue::gqs-receive-items" on page 365

"gqs-queue::gqs-receive-items" on page 370

gqs-remove-all-monitored-attributes

Synopsis

gqs-remove-all-monitored-attributes
(*Queue*: gqs-queue, *Client*: object

Argument	Description
Queue	The queue that is the target of this call.
Client	The client for this call.

Description

This procedure removes all attributes from the set of monitored attributes. This procedure does not turn off monitoring, so if attributes are added via gqs-add-monitored-attributes, monitoring already will be on. To deactivate monitoring, use "gqs-deactivate-attribute-monitoring" on page 189.

gqs-remove-items

Synopsis

gqs-remove-items
(*Queue*: gqm-queue, *ItemsToRemove*: item-list, *Client*: object)

Argument	Description
Queue	The queue containing items to be removed.
ItemsToRemove	A list of items to be removed from *Queue*.
Client	The client for this call.

Description

Use this method to remove a list of items from a queue. See also "gqs-remove-single-item" on page 200.

For the GQM queues (in contrast to the underlying gqs-queue behavior), removing an entry from the only queue on which it resides also deletes that entry. The deletion is carried out in a separate thread. This enables any updating of queues which is to take place in the current thread to complete before the actual deletions occur. Deletion does not occur if the entry resides on any gqm-queue in addition to *Queue*.

The class gda-alarm-queue does not remove unacknowleged entries. If an alarm entry (class gda-alarm-entry) is unacknowleged, that entry remains on the queue. The relation a-gda-active-entry-of is broken so that the entry is no longer treated as a valid entry for future alarm transitions.

If you want to add side effects when items are removed from a queue, you should subclass "gqm-queue" on page 170 (or the appropriate subclass), and override this method. Be sure to use a call next method statement in your method.

Applicable methods

"gda-alarm-queue::gqs-remove-items" on page 352

"gqm-queue::gqs-remove-items" on page 366

gqs-remove-monitored-attributes

Synopsis

gqs-remove-monitored-attributes
(*Queue*: gqs-queue, *Attributes*: symbol-list, *Client*: object)

Argument	Description
Queue	The queue that is the target of this call.
Attributes	A list of attributes that are to be monitored for changes.
Client	The client for this call.

Description

Use this procedure to remove attributes that are currently in the set of monitored attributes of the queue. See also "gqs-add-monitored-attributes" on page 187 and "gqs-remove-all-monitored-attributes" on page 197.

gqs-remove-single-item

Synopsis

gqs-remove-single-item
(*Queue*: gqs-queue, *Item*: item, *Client*: object)

Argument	Description
Queue	The queue that contains the item to be removed.
Item	The item to be removed from *Queue*.
Client	The client for this call.

Description

Use this procedure to remove an item from a queue, without an explicit sender. If you need to remove multiple items from a queue at one time, it is more efficient to use "gqs-remove-items" on page 198.

gqs-send-items

Synopsis

gqs-send-items
(*Sender*: gqs-queue, *Receiver*: gqs-queue, *SendList*: item-list,
Client: object)

Argument	Description
Sender	The queue responsible for sending items to the receiving queue.
Receiver	The queue receiving the items in *SendList*.
SendList	The list of items to be added to *Receiver*.
Client	The client for this call.

Description

Use this procedure to send a list of items to a queue. This procedure provides an efficient way to send multiple items from one queue to another. In addition, this procedure insures compatible queue and entry types when entries are sent from one queue to another.

gqs-send-single-item

Synopsis

gqs-send-single-item
(*Sender*: item, *Receiver*: gqs-queue, *Item*: item, *Client*: object)

Argument	Description
Sender	The source of the item.
Receiver	The queue receiving the item.
Item	The item being sent to *Receiver*.
Client	The client for this call.

Description

Use this procedure to send a single item to a queue. If you need to send multiple items to a queue at one time, it is more efficient to use "gqs-send-items" on page 201. If the sender is a gqs-queue, this procedure has the same effect as calling gqs-send-items with a list of one item. If the sender is not a gqs-queue, this method is the same as using gqs-receive-single-item on the *Receiver*.

Entry Classes

Class Hierarchy

Each type of queue has a unique class of entry that populates it. The class of entry that a queue generates can be overridden by changing the gqm-queue attribute gqm-entry-class. You can create new classes of entry and use them in the built-in queues.

This built-in queue...	Generates entries of this class...
Alarm queue	gda-alarm-entry
Error queue	gqm-error-entry
Explanation queue	gda-explanation-entry
Message queue	gqm-entry

An application programmer might subclass entries for several reasons. If extra columns are to be displayed in the queue views, it is probably necessary to add extra attributes to the entry class to hold the data for those columns. Additionally, extensions of the methods on gqm-entry and its subclasses may be required. In general, if these two functions are not met, it may be easier to accomplish the same functionality by extending gqm-queue class, the gqs-view-manager class, or the gqm-logging-manager class.

gda-alarm-entry

This is the class used by the built-in alarm queue.

The gda-alarm-entry is created, like gqm-entry, by calls to gqm-post-entry. However, there is also a second, similar, call gda-update-existing-alarm-entry. Alarm entries are updated when the source goes out of alarm. Also if the gda-alarm-queue-setting attribute gda-reuse-entry is true, an existing alarm entry continues to be reused as the entry goes in and out of alarm until that entry is acknowledged.

This class also has a different method for customizing alarm logging. Refer to the gda-alarm-queue-setting attribute gda-alarm-log-formatter. But before extending the gda-alarm-entry class to customize alarm logging, read the documentation for "gqm-logging-manager" on page 251.

In addition to the attributes listed below, there is a gda-alarm-status associated with alarm entries. This attribute does not really exist on the gda-alarm-entry class. It is defined, however, as a "virtual attribute," which is used by the attribute update mechanism. If you have a gqs-view-manager that is listening to changes to the queue, it is updated (via gqs-view-manager::gqs-update-view-per-attribute) with changes in the status of the entry. Likewise, if a gqs-attribute-update-callback procedure has been configured for the gda-alarm-queue on which this entry resides, that procedure is called when this virtual attribute is changed.

Changes in the gda-alarm-status virtual attribute indicate changes in one of two pieces of data associated with this alarm. Either the alarm-status attribute of the gdl-alarm-source associated with this entry has changed, or the relation a-gda-active-entry-of has been broken. In the built-in views, a change in this virtual attribute indicates that the entry display should change color.

Class Inheritance Path

gda-alarm-entry, gda-entry-with-uuid, gqm-entry, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
gda-require-acknowledgement	This attribute takes its value from the gdl-alarm-source attribute require-acknowledgement of the source of *Entry*. See "gda-alarm-entry::gqm-entry-constructor" on page 350 for more details. This attribute is read-only. To change the require-acknowledgement of a gdl-alarm-source, configure the alarm source directly.
Allowable values:	Any truth-value
Default value:	true
acknowledged	This entry mirrors the acknowledged attribute on gdl-alarm-source. Typically, this attribute is read-only. To programmatically acknowledge an alarm via its entry, use "gqsv-acknowledge" on page 224. If you have a gqs-view-manager listening to changes to the queue, it is updated (via gqs-view-manager::gqs-update-view-per-attribute) whether or not that view is monitoring for attribute changes. Likewise, if a gqs-attribute-update-callback procedure has been configured for the gda-alarm-queue on which this entry resides, that procedure is called when this attribute is changed.
Allowable values:	Any truth-value
Default value:	false
gda-acknowledge-time	This attribute stores the time, in G2 time, when the alarm associated with the entry was acknowledged. It then is used when generating the log for that entry.
Allowable values:	Any float
Default value:	0.0

Attribute	Description
gda-severity	This attribute mirrors the severity of the associated alarm source. It is passed to the alarm indirectly through gqm-post-entry. The attribute itself should be considered to be read-only.
Allowable values:	Any integer
Default value:	0
gdl-filter-tag	This attribute mirrors the gdl-filter-tag of the associated gdl-alarm-source. This attribute should not be written directly. It provides an additional attribute used for sorting or filtering alarms.
Allowable values:	Any symbol
Default value:	UNSPECIFIED
gqm-comments	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	TEXT-ARRAY
gqm-message-text	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	""
gqm-creation-time	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	0.0
gqm-priority	See "gqm-entry" on page 213.

Attribute	Description
Allowable values:	Inherited
Default value:	0

Extensible Methods

"gda-alarm-entry::gda-update-existing-alarm-entry" on page 349
"gda-alarm-entry::gqm-entry-constructor" on page 350
"gda-alarm-entry::gqm-expire-entry" on page 351
"gda-entry-with-uuid::gqm-entry-constructor" on page 353
"gqm-entry::gqm-delete" on page 358
"gqm-entry::gqm-entry-constructor" on page 359
"gqm-entry::gqm-expire-entry" on page 360
"gqm-entry::gqm-log-entry" on page 361
"gqm-entry::gqm-save-entry" on page 362

Relations

"a-gda-active-entry-of" on page 388

gda-explanation-entry

The purpose of this subclass is to customize the logging and saving methods. Also, entries of this class expire if the gqm-queue attribute gqm-entry-lifetime is greater than zero.

Gda-explanation-entry is the default entry class for the explanation queue.

Class Inheritance Path

gda-explanation-entry, gda-entry-with-uuid, gqm-entry, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
gqm-comments	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	TEXT-ARRAY
gqm-message-text	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	""
gqm-creation-time	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	0.0
gqm-priority	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	0

Extensible Methods

gda-recurring-alarm-entry

This subclass of gda-alarm-entry has one additional attribute, the number of recurrences.

Entries of this class are posted to the alarm queue by instances of gdl-recurring-alarm. Typically they are not created programmatically. However, calling gqm-post-entry with a gdl-recurring-alarm named as the alarm source creates an entry of this class.

Class Inheritance Path

gda-recurring-alarm-entry, gda-alarm-entry, gda-entry-with-uuid, gqm-entry, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
gda-recurrances	The number of times the associated alarm has entered alarm. (The attribute is misspelled in the code.)
Allowable values:	Any integer
Default value:	0
gda-require-acknowledgement	See "gda-alarm-entry" on page 204.
Allowable values:	Inherited
Default value:	true
acknowledged	See "gda-alarm-entry" on page 204.
Allowable values:	Inherited
Default value:	false
gda-acknowledge-time	See "gda-alarm-entry" on page 204.

Attribute	Description
Allowable values:	Inherited
Default value:	0.0
gda-severity	See "gda-alarm-entry" on page 204.
Allowable values:	Inherited
Default value:	0
gdl-filter-tag	See "gda-alarm-entry" on page 204.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqm-comments	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	TEXT-ARRAY
gqm-message-text	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	""
gqm-creation-time	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	0.0
gqm-priority	See "gqm-entry" on page 213.

Attribute	Description
Allowable values:	Inherited
Default value:	0

Extensible Methods

gqm-entry

This is the basic class of all entries in the queue system. It is also the class used on the built-in queue instance message queue.

An entry is created using the call gqm-post-entry on a queue. Gqm-post-entry takes a structure that is passed on to the gqm-entry-constructor method on the entry which is created. Refer to the documentation of "gqm-post-entry" on page 183 which gives the attributes on that structure which are defined for the built-in entry classes. This same structure is passed on to the gqm-entry-constructor method.

Class Inheritance Path

gqm-entry, object, item

Attributes

Attribute	Description
gqm-comments	This attribute can be accessed directly by an application programmer to add a comment programmatically.
Allowable values:	Any text-array
Default value:	TEXT-ARRAY
gqm-message-text	The text of the message, as it appears on the tabular display.
Allowable values:	Any text
Default value:	""
gqm-creation-time	A time stamp indicating when the entry was created.
Allowable values:	Any float
Default value:	0.0

Attribute	Description
gqm-priority	The priority is an integer that is not used by the rest of the queue system. It provides an extra attribute that application programmers can make use of. An entry has a priority which corresponds to the gqm-queue attribute gqm-default-priority of the queue that created it.
Allowable values:	Any integer
Default value:	0

Extensible Methods

"gqm-entry::gqm-delete" on page 358

"gqm-entry::gqm-entry-constructor" on page 359

"gqm-entry::gqm-expire-entry" on page 360

"gqm-entry::gqm-log-entry" on page 361

"gqm-entry::gqm-save-entry" on page 362

Relations

the-gqmv-detail-view-of

the-gqm-source-of

gqm-error-entry

The purpose of this subclass is to provide specific methods for gqm-error-entry::gqm-delete and gqm-error-entry::gqm-entry-constructor. There are no class-specific attributes on this class.

Gqm-error-entry is the default entry class for the error queue.

Class Inheritance Path

gqm-error-entry, gqm-entry, object, item

Attributes

Attribute	Description
gqm-comments	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	TEXT-ARRAY
gqm-message-text	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	""
gqm-creation-time	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	0.0
gqm-priority	See "gqm-entry" on page 213.
Allowable values:	Inherited
Default value:	0

Extensible Methods

Entry Procedures

This section describes the procedures that support the queue entries.

gda-get-explanation

Synopsis

gda-get-explanation
(*Entry*: gda-alarm-entry, *Client*: ui-client-item)
–> *Sequence*

Argument	Description
Entry	The alarm entry whose explanation is to be returned.
Client	The client or window.

Return Value	Description
Sequence	A sequence of texts, each being an explanation entry for the alarm.

Description

Returns the explanation of an alarm entry.

gda-get-history

Synopsis

gda-get-history
(*Entry*: gda-alarm-entry, *Client*: ui-client-item)
–> *Sequence*

Argument	Description
Entry	The alarm entry whose history is to be returned.
Client	The client or window.

Return Value	Description
Sequence	A sequence of texts, each being a history entry for the alarm.

Description

Returns the history of an alarm entry.

gda-update-existing-alarm-entry

Synopsis

gda-update-existing-alarm-entry
(*Entry*: gda-alarm-entry, *AlarmData*: structure, *Client*: ui-client-item)

Argument	Description
Entry	An existing alarm entry that is to be updated.
AlarmData	See below for a description of this structure.
Client	The client for the call.

Description

This procedure is very similar to gqm-post-entry, but for alarm entries only. Whereas gqm-post-entry is an API for creating a new entry, given a set of data for the alarm, this procedure is used to update an existing entry, given a change in state in the alarm. The nature of the state change must be included in the call as the gda-transition of *AlarmData*.

AlarmData Structure Description

The required attributes in the *AlarmData* data structure are a superset of those required for gqm-post-entry.

Attribute	Type	Description
gda-transition	Symbol	A symbol, one of: NEW-ALARM RETURN-TO-ALARM RETURN-FROM-ALARM RETURN-FROM-RECURRING-ALARM CONTINUE-RECURRING-ALARM The last two only apply to recurring alarms and generally are not passed in a call to this API.
gqm-text	Text	The text of the alarm. The text of an existing alarm entry is not replaced by the new text. This attribute is only used if a new entry must be posted.

Attribute	Type	Description
gqm-source	gdl-Alarm-Source	The source of the alarm.
gda-status	Symbol	A status associated with the belief (below). The status is either .TRUE, .FALSE, or UNKNOWN. See "gqm-post-entry" on page 183 for more information about this attribute.
gda-belief	Float	The fuzzy belief value of the alarm, with value from 0.0 to 1.0. The old belief is not replaced in the new text, if applicable.
gda-collection-time	Float	The timestamp for the data that caused the alarm.
gda-severity	Integer	The severity of the alarm

Applicable methods

gda-Alarm-Entry::gda-Update-Existing-Alarm-Entry

gda-Recurring-Alarm-Entry::gda-Update-Existing-Alarm-Entry

gqm-expire-entry

Synopsis

gqm-expire-entry
(*Entry*: gdm-entry, *Queue*: gqm-queue, *Source*: item, *Client*: object)

Argument	Description
Entry	The entry to expire.
Queue	The queue on which the entry was posted. If *Entry* is a gda-alarm-entry, then *Queue* must be a gda-alarm-queue.
Source	The source of *Entry*, if applicable. If *Entry* is a gqm-entry, then any item (e.g., *Client*) can be passed into this argument; it is not used. If *Entry* is a gda-alarm-entry, then *Source* must be a gda-alarm-source.
Client	The client for this call.

Description

You should call gqm-expire-entry from an entry constructor. The procedure waits for the lifetime specified on the gqm-queue attribute gqm-entry-lifetime and then removes *Entry* from all queues on which it resides. If there is a detail view of the entry at the time the expiration is to take place, it waits for the detail view to be deleted. The Update Latency of the (original) queue is used to determine the polling frequency.

If *Entry* is a gda-alarm-entry, then gqm-expire-entry additionally logs an expiration event to the history, then removes the entry from all queues (alarm or otherwise).

Applicable methods

gda-Alarm-Entry::gqm-Expire-Entry

gqm-Entry::gqm-Expire-Entry

gqm-save-entry

Synopsis

gqm-save-entry
(*Entry*: gqm-entry, *Queue*: gqm-queue, *Filename*: text, *Client*: object)

Argument	Description
Entry	The entry to save.
Queue	The queue initiating the save.
Filename	The file to which to save. File formats are documented in the *G2 System Procedures Reference Manual*.
Client	The client for the call.

Description

This procedure initiates the save of the contents of *Entry*. This is the procedure called by the gqmv-save-selected-button of a view of *Queue,* but it can also be accessed directly.

Applicable methods

gda-Alarm-Entry::gqm-Save-Entry

gqm-Entry::gqm-Save-Entry

gqsv-acknowledge

Synopsis

gqsv-acknowledge
(*Entry*: gda-alarm-entry, *Client*: ui-client-item)

Argument	Description
Entry	The entry whose source is to be acknowledged.
Client	The client for this call.

Description

This method provides the ability to acknowledge an alarm, given an entry. The API call is necessary because it is the alarm source that is really acknowledged within GDA, not the entry itself. The attribute acknowledged on *Entry* should not be modified directly, as it will not propagate its value correctly to the alarm source, alarm panels and related items.

This is the API that the gqsv-acknowledge-button calls.

Entry Sources

This section describes the class definitions for alarm and message source objects.

gdl-alarm

This is the class definition of the palette item referred to as Alarm or Alarm Capability. You can access instances of this class directly from your diagrams or as the object of the entry relations the-gqm-source-of or the-gda-active-alarm-of.

This class should not be subclassed.

Class Inheritance Path

gdl-alarm, gdl-alarm-capability, gdl-capability, gdl-block, gdl-task, gdl-task-or-event, gdl-alarm-source, gdl-object, gfr-object-with-uuid, gdl-object-with-sse-id, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
belief	This read-only attribute is the truth or falseness of the alarm, expressed as a fuzzy value. In binary logic, 0.0 is false and 1.0 is true.
Allowable values:	Any item or value
Default value:	0.0
memory-enabled	This attribute is read-only. A value of true indicates that the alarm has been configured to have memory enabled.
Allowable values:	Any truth-value
Default value:	true
update-alarm-readouts	Setting this value to true indicates that the color of the alarm should be reflected in any alarm readouts.
Allowable values:	Any truth-value
Default value:	true

Attribute	Description
acknowledged	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	false
alarm-status	See "gdl-alarm-source" on page 233.
Allowable values:	NO-ALARM, INHIBITED, IN-ALARM, ALARM-IN-MEMORY
Default value:	NO-ALARM
severity	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	1
trigger-on	See "gdl-alarm-capability" on page 230.
Allowable values:	.TRUE, .FALSE, UNKNOWN
Default value:	.TRUE
show-messages	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	true
require-acknowledgement	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	true

Attribute	Description
generate-automatic-explanation	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	false
gdl-display-queue	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	ALARM-QUEUE
advice	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	""
gdl-filter-class	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gdl-filter-tag	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	UNSPECIFIED
error	See "gdl-object" on page 80.
Allowable values:	Inherited
Default value:	""
comments	See "gdl-object" on page 80.

Attribute	Description
Allowable values:	Inherited
Default value:	NONE

Relations

gda-color-display-for

gdl-alarm-capability

This class is a superior for both the gdl-alarm and gdl-recurring-alarm. It should neither be instantiated nor subclassed.

Class Inheritance Path

gdl-alarm-capability, gdl-capability, gdl-block, gdl-task, gdl-task-or-event, gdl-alarm-source, gdl-object, gfr-object-with-uuid, gdl-object-with-sse-id, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
acknowledged	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	false
alarm-status	See "gdl-alarm-source" on page 233.
Allowable values:	NO-ALARM, INHIBITED, IN-ALARM, ALARM-IN-MEMORY
Default value:	NO-ALARM
severity	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	1
trigger-on	Which status value of the associated block causes the alarm to fire.
Allowable values:	.TRUE, .FALSE, UNKNOWN
Default value:	.TRUE

Attribute	Description
show-messages	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	true
require-acknowledgement	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	true
generate-automatic-explanation	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	false
gdl-display-queue	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	ALARM-QUEUE
advice	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	""
gdl-filter-class	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	UNSPECIFIED

Attribute	Description
gdl-filter-tag	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	UNSPECIFIED
error	See "gdl-object" on page 80.
Allowable values:	Inherited
Default value:	""
comments	See "gdl-object" on page 80.
Allowable values:	Inherited
Default value:	NONE

gdl-alarm-source

This is the foundation class for all alarm sources within GDA. This class should not be directly instantiated; however, instances of gdl-alarm and gdl-recurring-alarm are also of this class. This class should be used, as a mix-in class, by the application developer who wants to have any G2 object class serve as a GDA alarm source.

If the class instance is a built-in capability (one supplied with GDA), configuration is done typically through the configuration dialog by the user. If you have subclassed gdl-alarm-source, you must provide the mechanism for configurating the instances. Except where indicated, all attributes defined by gdl-alarm-source can be written (to configure) and can be read.

Class Inheritance Path

gdl-alarm-source, gdl-object, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
acknowledged	This attribute should not be set directly. Instead, use the API "gda-acknowledge-alarm" on page 242. When read, a value of true indicates that this alarm has been acknowledged.
Allowable values:	Any truth-value
Default value:	false

Attribute	Description
alarm-status	Normally, this attribute is read-only. The alarm status is the attribute where the gdl-alarm-source keeps track of its internal state. The individual values of this attribute imply the following states for the alarm: NO-ALARM - alarm is in its reset state. No alarm condition exists. INHIBITED - the alarm has been inhibited. Changes in the associated GDA blocks do not affect the state of the alarm. IN-ALARM - the conditions that trigger the alarm have occurred. For any subclass that does not implement the more complex behaviors of histories, this would be the normal value for an alarm associated with an active gda-alarm-entry (see "gdl-queue-message" on page 237). ALARM-IN-MEMORY - for an alarm that is keeping history, the alarm has gone into alarm and returned from alarm without the alarm having been acknowledged. When the alarm returns from alarm, instead of the status changing to NO-ALARM, it becomes ALARM-IN-MEMORY. If the alarm has been acknowledged, the alarm returns to NO-ALARM. Alarm memory is configured on gdl-alarm. Changes in alarm status are directly related to the gda-transition described in the "gda-update-existing-alarm-entry" on page 220 API.
Allowable values:	NO-ALARM, INHIBITED, IN-ALARM, ALARM-IN-MEMORY
Default value:	NO-ALARM
severity	A configurable attribute that enables the user to classify alarms at different levels. The severity of an alarm is normally also reflected in the color used to highlight its associated entries and readouts.
Allowable values:	Any integer

Attribute	Description
Default value:	1
show-messages	A configurable attribute that indicates whether or not a view of the alarm queue should be launched when this gdl-alarm-source goes into alarm.
Allowable values:	Any truth-value
Default value:	true
require-acknowledgement	A configurable attribute that indicates whether or not user acknowledgement of alarms are required before they can be removed from the system. Alarms that do not require acknowledgment are created with their acknowledged attribute initially set to true.
Allowable values:	Any truth-value
Default value:	true
generate-automatic-explanation	Indicates whether an explanation for the alarm (see "gda-get-explanation" on page 218) should be generated automatically when alarm condition occurs.
Allowable values:	Any truth-value
Default value:	false
gdl-display-queue	The symbol names a gda-alarm-queue instance on which to post entries for the alarm.
Allowable values:	Any symbol
Default value:	ALARM-QUEUE
advice	Configure this attribute to contain advice to the user on what action to take when the alarm occurs. See the *GDA User's Guide*.

Attribute	Description
Allowable values:	Any text
Default value:	""
gdl-filter-tag	This is a user-configurable attribute (not used by GDA) provided for application programmers to use to customize the queue system. It is envisioned as an attribute to be used in conjunction with filtering or subscription.
Allowable values:	Any symbol
Default value:	UNSPECIFIED
error	See "gdl-object" on page 80. This attribute is typically read-only.
Allowable values:	Inherited
Default value:	""
comments	See "gdl-object" on page 80.
Allowable values:	Inherited
Default value:	NONE

Extensible Methods

"gdl-alarm-source::gda-acknowledge-alarm" on page 356

"gdl-object::gdl-clear-error" on page 110

"gdl-object::gdl-configure" on page 111

"gdl-object::gdl-get-configuration" on page 113

"gdl-object::gdl-initialize" on page 114

Relations

a-gda-active-entry-of

gdl-queue-message

The gdl-queue-message block is designed to post messages to a message queue (see "gqm-queue" on page 170). You could also post messages to a message queue directly with "gqm-post-entry" on page 183.

The gdl-queue-message block is not limited to posting to the message queue. In fact, you can post to any of the queue classes within GDA. In order to support this, the Queue Message block has attributes that enable it to generate alarm entries. Specifically, the attributes gdl-belief-value and gdl-belief-status, described in the following table, support data that would normally be provided through an associated Alarm Capability, but in this case must be configured directly on the Queue Message block.

If you are posting to any queue except the alarm queue, then the alarm attributes are not required. The attribute gdl-entry-text is the only configurable attribute which applies to the gqm-queue class.

Normally, this block would be referenced programmatically as the-gqm-source-of some gqm-entry. This class should not be subclassed.

Class Inheritance Path

gdl-queue-message, gdl-action, gdl-block, gdl-task, gdl-task-or-event, gdl-alarm-source, gdl-object, gfr-object-with-uuid, gdl-object-with-sse-id, gdl-entry-source, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
acknowledged	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	false
alarm-status	See "gdl-alarm-source" on page 233.
Allowable values:	NO-ALARM, INHIBITED, IN-ALARM, ALARM-IN-MEMORY
Default value:	IN-ALARM

Attribute	Description
severity	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	1
trigger-on	See "gdl-alarm-source" on page 233.
Allowable values:	.TRUE, .FALSE, UNKNOWN
Default value:	.TRUE
show-messages	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	true
require-acknowledgement	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	true
generate-automatic-explanation	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	false
gdl-display-queue	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	MESSAGE-QUEUE

Attribute	Description
advice	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	""
gdl-filter-class	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gdl-filter-tag	See "gdl-alarm-source" on page 233.
Allowable values:	Inherited
Default value:	UNSPECIFIED
error	See "gdl-object" on page 80.
Allowable values:	Inherited
Default value:	""
comments	See "gdl-object" on page 80.
Allowable values:	Inherited
Default value:	NONE
gdl-entry-text	The text of the message that is to be posted. It is required to post to any gqm-queue. If this attribute is left as an empty string (the default), no message is posted.
Allowable values:	Any text

Attribute	Description
Default value:	""
gdl-belief-value	This value corresponds to the gdl-alarm attribute belief. In the case of the gdl-queue-message, however, this attribute is configurable by the user.
Allowable values:	Any float
Default value:	0.5
gdl-belief-status	This attribute is configurable by the user. The value should correspond to the gdl-belief-value. For the gdl-alarm, this value is computed internally as described in the Fuzzy Logic section of the *GDA User's Guide*.
Allowable values:	.TRUE, .FALSE, UNKNOWN
Default value:	UNKNOWN

Procedures for Alarm Source

This procedure applies to entry sources.

gda-acknowledge-alarm

Synopsis

gda-acknowledge-alarm
(*Alarm*: gdl-alarm-source, *AckTime*: float, *Client*: ui-client-item)

Argument	Description
Alarm	The alarm to acknowledge.
AckTime	The time of acknowledgement, in normal G2 time.
Client	The client or window.

Description

Acknowledges an alarm source. This also has the side effect of updating every entry and calling the gda-log-alarm-summary method for each entry that is an active entry of the alarm.

Filters, Subscription, and Logging

This section describes the API objects pertaining to filtering, subscription, and logging. One common thread between these topics is that they have substantial API components. A second common thread is that these features are all exposed from the underlying modules (hence the APIs often contain GLF or GQS prefixes) and have very little in the way of customization for GDA.

The operations described here can often be accomplished in several different ways. For example, a problem that could be solved using filters and the view filtering buttons might also be solved using subscriptions and filters.

Another example is that a problem that might be considered one for logging could also be solved (more efficiently) using the gqs-view-manager. Suppose you want to log data about alarm occurrences and acknowledgements to a data base. Initially, this might seem like something the logging managers should be customized to do. However, by using the gqs-view-manager, it would be much easier to get notifications of changes to the queues and to send that event elsewhere in your system, packaging the data as needed.

Filters

Three instantiable filter classes are provided: gqs-attribute-filter, gqs-and-filter, and gqs-or-filter. The latter two enable two or more instances of the first to be combined into a single algorithm. In addition, the parent class, gqs-filter, is subclassable to enable you to define any arbitrary filtering algorithm that can then be used with the queues.

Subscriptions

The description of filtering in the documentation on queues in the *GDA User's Guide* refers to a view-based filter. With subscriptions, the filters can be applied between two queues. Subscribing queues could also be used, with or without filters, to route entries from multiple queues to a central queue for a single display. For example, multiple alarm queues might be defined to generate different entry classes or have different expiration times, but then a single, subscribing queue be used to collect all the different entries in a single place for display.

Logging

Logging managers provide an easy way to manage the logging of entry data to files. Logging managers use the G2 system procedures for opening, closing, and writing to files, but in a way that manages it out of sight of the user.

There are two classes of logging manager within GDA applications.

- Gqm-logging-manager corresponds to the three built-in queues of class gqm-queue.
- Gda-alarm-logging-manager contains additional attributes that enable configuration of alarm logging.

Typically, you do not need to subclass these any further. The attributes on the logging manager, as well as the gda-alarm-queue-setting attribute gda-alarm-log-formatter, can be set to your own procedures, enabling necessary customization. At any time, you can call glf-write-to-log-file within your code to add to a log.

Logging is initiated from the queues. The command whether to log or not, or the configurations of logging are all queue-based. However, the details of how logging is formatted is contained in the log-entry method on the class gqm-entry. See the documentation for "gqm-entry" on page 213 and "gda-alarm-entry" on page 204 for instructions on modifying the format and content of logged events.

Note that view manager provides a nice way to subscribe to events. At that point, you can log them, perhaps in a way unrelated to how the logging manager is set up. For example, if you are logging events to a database, it might be a better solution to access the database from a subscribed view manager rather than through a queue or a logging manager.

gda-alarm-logging-manager

This class is specialized to log the additional attributes important to GDA alarms. Instances of this class of logging manager should be used with instances of gda-alarm-queue.

Class Inheritance Path

gda-alarm-logging-manager, gqm-logging-manager, glf-logging-manager, object, item

Attributes

Attribute	Description
gda-log-events-incrementally	A configurable attribute indicating whether or not to log each time an alarm enters or leaves its alarm state.
Allowable values:	Any truth-value
Default value:	false
gda-log-explanations	A configurable attribute indicating whether explanations should be included in the log. See "gdl-alarm-source" on page 233. and "gda-get-explanation" on page 218.
Allowable values:	Any truth-value
Default value:	false
gda-log-advice	A configurable attribute indicating whether advice should be included in the log.
Allowable values:	Any truth-value
Default value:	false
gqm-time-format	See "gqm-logging-manager" on page 251.
Allowable values:	DATE, TIME, DATETIME, FILETIME

Attribute	Description
Default value:	DATETIME
gqm-log-comments	A configurable attribute indicating whether comments should be included in the log. It is also used by gqm-save-entry (see "gqm-logging-manager" on page 251).
Allowable values:	Inherited
Default value:	false
glf-logging-enabled	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	false
glf-log-directory	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	""
glf-log-file-name-template	See "gqm-logging-manager" on page 251.
Allowable values:	Inherited
Default value:	"loga_*.txt"
glf-log-file-name-generator	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-FILE-NAME-GENERATOR

Attribute	Description
glf-current-log-file	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	""
glf-file-header-writer	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-LOG-FILE-HEADER-WRITER
glf-time-interval-to-open-new-log-file	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	86400
glf-maximum-file-size-in-bytes	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	100000
glf-log-file-scheduler	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-LOG-FILE-SCHEDULER
glf-automatically-delete-empty-log-files	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	true

glf-logging-manager

This is the basic foundation class for logging managers. This class should not be extended by the GDA user. It is included in the documentation for reference only.

Class Inheritance Path

glf-logging-manager, object, item

Attributes

Attribute	Description
glf-logging-enabled	This read-only attribute indicates whether logging is enabled or disabled for this logging manager. The value is toggled by the user through menu choices. See also the API functions "glf-enable-logging" on page 266 and "glf-disable-logging" on page 265.
Allowable values:	Any truth-value
Default value:	false
glf-log-directory	This attribute is configurable by the user. It indicates the directory where log files are to be stored. If no text is given (the default), the default G2 directory is used.
Allowable values:	Any text
Default value:	""
glf-log-file-name-template	See "gqm-logging-manager" on page 251.
Allowable values:	Any text
Default value:	"log_*.txt"
glf-log-file-name-generator	Names a procedure that generates filenames. The procedure must be of the format as described in "glf-default-file-name-generator" on page 263.

Attribute	Description
Allowable values:	Any symbol
Default value:	GLF-DEFAULT-FILE-NAME-GENERATOR
glf-current-log-file	This attribute is read-only. The name of the current log file is written here.
Allowable values:	Any text
Default value:	""
glf-file-header-writer	Names a procedure that generates the header for the log files. The procedure must be of the format as described in "glf-default-log-file-header-writer" on page 264.
Allowable values:	Any symbol
Default value:	GLF-DEFAULT-LOG-FILE-HEADER-WRITER
glf-time-interval-to-open-new-log-file	A configurable time interval, after which to open a new log file.
Allowable values:	Any integer
Default value:	86400
glf-maximum-file-size-in-bytes	The log file size which, when exceeded, causes the logging manager to open a new file. Units are in bytes.
Allowable values:	Any integer
Default value:	100000
glf-log-file-scheduler	The symbol names a procedure that determines when a newly started log file should be closed. Modifying this attribute overrides the default behavior. See "glf-default-log-file-scheduler" on page 396 for the proper format for this procedure.

Attribute	Description
Allowable values:	Any symbol
Default value:	GLF-DEFAULT-LOG-FILE-SCHEDULER
glf-automatically-delete-empty-log-files	A user-configurable attribute that determines whether, when creating a new file, old and empty files should be removed.
Allowable values:	Any truth-value
Default value:	true

gqm-logging-manager

Each queue optionally has a logging manager associated with it. The logging manager specifies the filename, the format of the file, and when to change files when creating a permanent log for queue events. This class of logging manager is used for all the built-in queues of class gqm-queue. Specifically, the message queue, error queue, and explanation queue use a logging manager of class gqm-logging-manager.

This class extends glf-logging-manager by adding two attributes (see table).

Class Inheritance Path

gqm-logging-manager, glf-logging-manager, object, item

Attributes

Attribute	Description
gqm-time-format	A symbol to be passed as the Form argument in gqm-time-to-text. This attribute enables you to customize whether the date, the time, or the date and the time is used in the log file when a time stamp is recorded. The class gqm-general-setting enables the details of the format to be specified.
Allowable values:	DATE, TIME, DATETIME, FILETIME
Default value:	DATETIME
gqm-log-comments	This configurable attribute is not used for logging for instances of this class (see "gda-alarm-logging-manager" on page 245) as comments do not exist at the time entries are logged (on creation). However, this attribute is accessed by the API gqm-save-entry to determine whether or not comments should be written when an entry is saved.
Allowable values:	Any truth-value
Default value:	false
glf-logging-enabled	See "glf-logging-manager" on page 248.

Attribute	Description
Allowable values:	Inherited
glf-log-directory	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	""
glf-log-file-name-template	Provides a template used to generate successive filenames. In GDA, a different template is used for each built-in queue type because the actual log filenames used are generated based on this attribute and the time. If all four built-in queues were to start logging simultaneously (as happens on startup), they would all write to the same file. For this reason, you should create a new template each time you create a new instance of a gqm-logging-manager.
Allowable values:	Any text
Default value:	"log_*.txt"
glf-log-file-name-generator	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-FILE-NAME-GENERATOR
glf-current-log-file	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	""
glf-file-header-writer	See "glf-logging-manager" on page 248.
Default value:	false
Allowable values:	Inherited

Attribute	Description
Default value:	GLF-DEFAULT-LOG-FILE-HEADER-WRITER
glf-time-interval-to-open-new-log-file	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	86400
glf-maximum-file-size-in-bytes	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	100000
glf-log-file-scheduler	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	GLF-DEFAULT-LOG-FILE-SCHEDULER
glf-automatically-delete-empty-log-files	See "glf-logging-manager" on page 248.
Allowable values:	Inherited
Default value:	true

gqs-and-filter

This is a compound filter that applies AND logic. See "gqs-compound-filter" on page 256 for details.

Class Inheritance Path

gqs-and-filter, gqs-compound-filter, gqs-filter, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
gqs-filters	See "gqs-compound-filter" on page 256.
Allowable values:	Inherited
Default value:	GQS-PERMANENT-ITEM-LIST
gqs-combination-logic	See "gqs-compound-filter" on page 256.
Allowable values:	AND, OR
Default value:	AND

Extensible Methods

"gqs-filter::gqs-apply-filter" on page 368

gqs-attribute-filter

This filter tests the value of an attribute of an item against a target value, using equals, not-equals, contains, and other comparison functions. You would use an attribute filter, for example, if you want to select alarms with priority less than 3 by setting the attribute-name to PRIORITY, test to LESS-THAN, and target-value to 3.

Class Inheritance Path

gqs-attribute-filter, gqs-filter, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
attribute-name	The name of the attribute to be tested.
Allowable values:	Any symbol
Default value:	G2
test	The test to be applied.
Allowable values:	EQUALS, DOES-NOT-EQUAL, CONTAINS, DOES-NOT-CONTAIN, GREATER-THAN, GREATER-THAN-OR-EQUAL-TO, LESS-THAN, LESS-THAN-OR-EQUAL-TO, EXISTS, DOES-NOT-EXIST
Default value:	EQUALS
target-value	The target value of the comparison.
Allowable values:	Any value
Default value:	0.0

Extensible Methods

“gqs-filter::gqs-apply-filter” on page 368

gqs-compound-filter

This class defines a filter that is a combination of other filters.

- If the combination logic is AND, an item passes the compound filter only if it passes all subfilters.
- If the combination logic is OR, an item passes the compound filter if it passes any subfilter.

In other words, AND logic acts like subfilters are in series, with fewer and fewer items passing through each filter, while OR logic acts as if the subfilters are in parallel, and the output is the total of all passed items. To add a subfilter to a compound filter, use "gqs-populate-compound-filter" on page 276. See also "gqs-and-filter" on page 254 and "gqs-or-filter" on page 258.

Class Inheritance Path

gqs-compound-filter, gqs-filter, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
gqs-filters	A list of subfilters in this compound filter.
Allowable values:	Any gqs-permanent-item-list
Default value:	GQS-PERMANENT-ITEM-LIST
gqs-combination-logic	The type of combination logic to be applied
Allowable values:	AND, OR
Default value:	AND

Extensible Methods

"gqs-filter::gqs-apply-filter" on page 368

gqs-filter

This is the parent class for view filters and subscription filters. It is an abstract class and should not be instantiated.

To create a new type of filter, subclass gqs-filter and define the method gqs-apply-filter.

See "gqs-filter::gqs-apply-filter" on page 368 for details. Also see "gqs-attribute-filter" on page 255, "gqs-compound-filter" on page 256, "gqs-and-filter" on page 254, and "gqs-or-filter" on page 258.

Class Inheritance Path

gqs-filter, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

none

Extensible Methods

"gqs-filter::gqs-apply-filter" on page 368

gqs-or-filter

This is a compound filter that applies OR logic. See "gqs-compound-filter" on page 256 for details.

Class Inheritance Path

gqs-or-filter, gqs-compound-filter, gqs-filter, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
gqs-filters	See "gqs-compound-filter" on page 256.
Allowable values:	Inherited
Default value:	GQS-PERMANENT-ITEM-LIST
gqs-combination-logic	See "gqs-compound-filter" on page 256.
Allowable values:	AND, OR
Default value:	OR

Extensible Methods

"gqs-filter::gqs-apply-filter" on page 368

gqs-subscription

Use an instance of a gqs-subscription if you want one queue to send items that match certain filter criteria to another queue. A subscription always involves exactly two queues, a sender and receiver. When the receiver subscribes to the sender, it checks any items it subsequently receives against the filter criteria, and sends the appropriate items to the receiving queue. Generally, subscriptions are created programmatically using the create action. See "gqs-subscribe" on page 277, "gqs-unsubscribe" on page 278, and "gqs-attach-filter-to-subscription" on page 270 for more information.

Class Inheritance Path

gqs-subscription, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
gqs-send-already-collected-items	If this attribute is true, when the receiving queue subscribes to a sending queue, the sending queue immediately sends any items contained in the sending queue that match the filter criteria.
Allowable values:	Any truth-value
Default value:	false

Procedures for Filters, Subscription, and Logging

This section describes the procedures for filters, subscription, and logging.

gda-format-alarm-log-text

Synopsis

gda-format-alarm-log-text
(*LogActivity*: symbol, *LogMgr*: gda-alarm-logging-manager,
Entry: gda-alarm-entry, *Queue*: gda-alarm-queue, *Client*: ui-client-item)
–> *Text*

Argument	Description
LogActivity	A symbol indicating the activity to log. See the list below for the valid symbols and their meanings.
LogMgr	The logging manager for *Queue*.
Entry	The entry to log.
Queue	The queue that is logging *Entry*.
Client	The client for this call.

Return Value	Description
Text	The formatted log text corresponding to *LogActivity*.

Description

This procedure is named in the gda-alarm-log-formatter attribute of the active gda-alarm-queue-setting. To change the text that is written to an alarm log, write a new procedure with the same arguments as gda-format-alarm-log-text.

Example

This procedure can also be called as part of your custom log formatter. For example, if you want to change only the format of the ACKNOWLEDGED log entries, but keep all else the same, you could write the following procedure:

```
custom-format-log-text (LogActivity: symbol,
    LogMgr: class gda-alarm-logging-manager, Entry: class gda-alarm-entry,
    Queue: class gda-alarm-queue. Client: class ui-client-item)
    = (text)
FormattedText: text;
begin
```

```
        if LogActivity = the symbol ACKNOWLEDGED then return "This custom
            alarm was acknowledged on [the gda-acknowledged-time of Entry]";
        FormattedText = call gda-format-log-activity(LogActivity, LogMgr, Entry,
            Queue, Client);
        return FormattedText;
    end;
```

The activity symbols and their meanings are listed below:

ACKNOWLEDGED: an alarm or alarm entry was acknowledged

ADVICE: the advice of an alarm entry

COMMENTS: the entry comments

EXPLANATION: the explanation of the alarm

HEADER: the entry header

HISTORY: the history of an alarm entry

HISTORYPURGE: the history of an alarm entry was purged because the history limit was exceeded

INCREMENTAL: alarm event such as returning from alarm state or returning to alarm state

MESSAGETEXT: the message text of the entry

METADATA: creation and collection time data

SUMMARYHEADER: the header for the alarm summary section

UNACKNOWLEDGED: an alarm entry removed from the queue was unacknowledged

glf-default-file-name-generator

Synopsis

glf-default-file-name-generator
(*Log*: glf-logging-manager, *Client*: object)
–> *Text*

Argument	Description
Log	The logging manager that needs to generate a new filename.
Client	The client for this call.

Return Value	Description
Text	The new filename.

Description

This procedure is named in the glf-file-name-generator attribute of *Log*. If this procedure is used, the log file is stored in your working directory and is named `log_YYYYMMDDHHMMSS.txt`, where

`YYYYMMDD` is the year, month, and day when the file is created.
`HHMMSS` is the hour, minutes, and seconds when the file is created.

To change the way filenames are generated, write a new procedure with the same arguments as glf-default-file-name-generator. It is suggested that you use the template attribute of *Log* to create the new name.

glf-default-log-file-header-writer

Synopsis

glf-default-log-file-header-writer
(*Log*: glf-logging-manager, *Client*: object)

Argument	Description
Log	The logging manager controlling this logging session.
Client	The client for this call.

Description

When called, this procedure writes a header to the file being logged by *Log*. This procedure should not normally be called by the user, however, as it is called during the normal operation of the glf-logging-manager. If you are writing your own log file header writer, use glf-write-log-to-file(Log, "", Client) to write the lines of your header.

glf-disable-logging

Synopsis

glf-disable-logging
(*Log*: glf-logging-manager, *Client*: object)

Argument	Description
Log	A logging manager that is to cease logging.
Client	The client for this call.

Description

Turns logging off for *Log*.

glf-enable-logging

Synopsis

glf-enable-logging
(*Log*: glf-logging-manager, *Client*: object)

Argument	Description
Log	A logging manager that is to start logging.
Client	The client for this call.

Description

Turns logging on for *Log*.

glf-set-fixed-log-closing-times

Synopsis

glf-set-fixed-log-closing-times
(*Log*: glf-logging-manager, *Times*: integer-list, *Client*: object)

Argument	Description
Log	The logging manager whose closing is to be scheduled.
Times	An integer list. Each element is a closing time.
Client	The client for this call.

Description

Schedules the closing of the named logging manager. Inputs times in minutes since midnight. This procedure checks, sorts and converts to seconds.

glf-write-to-log-file

Synopsis

glf-write-to-log-file
(*Log*: glf-logging-manager, *Text*: text, *CheckSize*: truth-value, *Client*: object)

Argument	Description
Log	The logging manager controlling the target log file.
Text	A line of text to be written by *Log*.
CheckSize	When true, checks the size of the log file after writing the file to see if a new file needs to be generated.
Client	The client for this call.

Description

This procedure writes a line of text to the log specified by *Log*. When using this procedure, GLF is managing the opening and closing of G2 streams and well as managing the maximum size and age of the log files.

The procedure checks the file size after it writes the file to see if a new file needs to be created. GLF allows the user to specify a maximum file size for the log. If a new log file needs to be created, the procedure closes the existing log file, then opens a new one. After the new file is opened, the logging manager calls the log file header writer. User-defined header writers should pass false as the *CheckSize* argument to avoid checking the file size when a new file is already in the process of being created. All other callers should pass true.

gqs-apply-filter

Synopsis

gqs-apply-filter
(*Filter*: gqs-filter, *SourceList*: item-list, *ResultList*: item-list, *Client*: object)

Argument	Description
Filter	The filter to be applied to *SourceList*.
SourceList	The items to be filtered.
ResultList	A list to contain the results, which are the items passed by the filter.
Client	The client for this call.

Description

This procedure tests the value of an attribute of an item against a target value, using equals, not-equals, contains, and other comparison functions. You would use an attribute filter, for example, if you want to select alarms with priority less than 3 by setting the attribute-name to PRIORITY, the test to LESS-THAN, and the target-value to 3. The filter is applied to each item in the *SourceList* and the items passing the filter criteria are appended to the *ResultList*, without clearing the *ResultList* first.

If *Filter* is a gqs-compound-filter, then each of the subfilters of *Filter* is applied and the result combined using AND or OR logic. See "gqs-compound-filter" on page 256 for more details.

Applicable methods

gqs-Attribute-Filter::gqs-Apply-Filter

gqs-Compound-Filter::gqs-Apply-Filter

gqs-Filter::gqs-Apply-Filter

gqs-attach-filter-to-subscription

Synopsis

gqs-attach-filter-to-subscription
(*Subscription*: gqs-subscription, *Filter*: gqs-filter, *Client*: object)

Argument	Description
Subscription	The subscription to be associated with *Filter*.
Filter	The filter to be associated with *Subscription*.
Client	The client for this call.

Description

Use this procedure to associate a filter with a subscription. Each subscription can have at most one filter (the effect of multiple filters can be achieved with a compound filter). To remove the filter from a subscription, use "gqs-detach-filter-from-subscription" on page 271.

gqs-detach-filter-from-subscription

Synopsis

gqs-detach-filter-from-subscription
(*Subscription*: gqs-subscription, *Client*: object)

Argument	Description
Subscription	The target subscription.
Client	The client for this call.

Description

Use this procedure to remove a filter from a subscription.

gqs-get-subscription-details

Synopsis

gqs-get-subscription-details
(*Subscription*: gqs-subscription, *Client*: object)
–> *Item-Or-Value*, *Item-Or-Value*, *Item-Or-Value*

Argument	Description
Subscription	Any subscription
Client	The client for this call.

Return Value	Description
Item-Or-Value	The subscribing queue, or false.
Item-Or-Value	The providing queue, or false.
Item-Or-Value	The filter associated with the subscription, or false.

Description

This procedure returns information about a given subscription, in particular, the subscribing and providing queues, and the filter, if any, associated with the subscription. If any of these attributes are not defined for *Subscription*, false is returned.

gqs-get-subscriptions-from-queue

Synopsis

gqs-get-subscriptions-from-queue
(*Queue*: gqs-queue, *ItemList*: item-list, *Client*: object)

Argument	Description
Queue	The queue that is the target of this call.
ItemList	An item list to hold the results of this call.
Client	The client for this call.

Description

This procedure returns a list of subscriptions that have been issued from *Queue*. The subscriptions are appended to *ItemList* without clearing the list first. See also "gqs-get-subscriptions-from-queue-to-queue" on page 274 and "gqs-get-subscriptions-to-queue" on page 275.

gqs-get-subscriptions-from-queue-to-queue

Synopsis

gqs-get-subscriptions-from-queue-to-queue
(*Subscriber*: gqs-queue, *Provider*: gqs-queue, *ItemList*: item-list,
Client: object)

Argument	Description
Subscriber	The subscribing queue.
Provider	The queue that is passing items to the subscriber queue.
ItemList	A list to contain the results of this call.
Client	The client for this call.

Description

This procedure returns the subscriptions associated with two queues, the recipient queue (*Subscriber*) and sending queue (*Provider*). The subscriptions are appended to *ItemList*.

gqs-get-subscriptions-to-queue

Synopsis

gqs-get-subscriptions-to-queue
(*Queue*: gqs-queue, *ItemList*: item-list, *Client*: object)

Argument	Description
Queue	The queue that is the target of this call.
ItemList	An item list to contain the results.
Client	The client for this call.

Description

This procedure returns a list of subscriptions that are currently held by a queue. The subscriptions are appended to *ItemList* without clearing the list first. See also "gqs-get-subscriptions-from-queue" on page 273 and "gqs-get-subscriptions-from-queue-to-queue" on page 274.

gqs-populate-compound-filter

Synopsis

gqs-populate-compound-filter
(*CompoundFilter*: gqs-compound-filter, *Filters*: item-list,
ClearFirst: truth-value, *Client*: object)

Argument	Description
CompoundFilter	A compound filter containing none or any number of subfilters.
Filters	A list of subfilters to add to the compound filter.
ClearFirst	A flag indicating whether existing subfilters should be removed from the compound filter before adding the filters contained in *Filters*.
Client	The client for this call.

Description

Use this procedure to add subfilters to a compound filter. Using the *ClearFirst* flag, you can specify if the existing subfilters (if any) are removed from *CompoundFilter* before adding the filters given in the *Filters* item list. If the item list is empty and the *ClearFirst* flag is true, this has the effect of clearing the compound filter.

gqs-subscribe

Synopsis

gqs-subscribe
(*Subscription*: gqs-subscription, *Provider*: gqs-queue,
Requestor: gqs-queue, *Client*: object)

Argument	Description
Subscription	The subscription to be added to the queues.
Provider	The queue that holds the subscription and provides items to the *Requestor*.
Requestor	The queue that receives items from the *Provider*.
Client	The client for this call.

Description

Use this procedure to add a subscription between two queues.

- If no filter is attached to the subscription, all items added to the provider queue are sent to the requestor queue. With a filter, only those items fulfilling the filter criteria are sent.
- If the attribute gqs-send-already-collected-items of the subscription is true, then calling gqs-subscribe results in immediate forwarding of all items in the provider queue, or those that meet the filter criteria, if any.

gqs-unsubscribe

Synopsis

gqs-unsubscribe
(*Provider*: gqs-queue, *Subscription*: gqs-subscription,
Subscriber: gqs-queue, *Client*: object)

Argument	Description
Provider	The queue that holds the subscription.
Subscription	The subscription that is to be removed.
Subscriber	The subscriber to the subscription.
Client	The client for this call.

Description

Use this procedure to remove a subscription between two queues.

gqsv-activate-view-filter

Synopsis

gqsv-activate-view-filter
(*Manager*: gqsv-tabular-view-manager, *Client*: object)

Argument	Description
Manager	The manager of the view.
Client	The client for this call.

Description

This procedure activates the current view filter, if any, and refreshes the view. The filtered items are temporarily removed from view, but are still present in the underlying queue. To specify a view filter, use "gqsv-set-view-filter" on page 282.

gqsv-deactivate-view-filter

Synopsis

gqsv-deactivate-view-filter
(*Manager*: gqsv-tabular-view-manager, *Client*: object)

Argument	Description
Manager	The manager for the view.
Client	The client for this call.

Description

This procedure deactivates the current view filter, and makes all items in the queue visible. The filter remains assigned to the view, and can be reactivated by calling "gqsv-activate-view-filter" on page 279.

gqsv-get-view-filter

Synopsis

gqsv-get-view-filter
(*Manager*: gqsv-tabular-view-manager, *Client*: object)
–> *Item-Or-Value*

Argument	Description
Manager	The manager for the view.
Client	The client for this call.

Return Value	Description
Item-Or-Value	The gqs-filter associated with the view, if any, or false.

Description

This procedure returns the current view filter, if any. If there is no view filter, this procedure returns false.

gqsv-set-view-filter

Synopsis

gqsv-set-view-filter
(*Manager*: gqsv-tabular-view-manager, *Filter*: item-or-value,
Client: object)

Argument	Description
Manager	The manager of the view.
Filter	An instance of a gqs-filter, or false.
Filename	The file to which to save.
Client	The client for this call.

Description

This procedure sets the current view filter. If the *Filter* argument is passed as false, any existing view filter is removed.

- If filtering is active, the filter is immediately activated.
- If view filtering is not currently active, the filter is set, but not activated.

A subsequent call to "gqsv-activate-view-filter" on page 279 is required to activate the filter.

View Manager Definitions

This section documents the gqs-view-manager and its subclass gqmv-tabular-view-manager. Although this represents essentially one object definition, it is in a separate section from the Tabular View section because it can be used independently of the tabular views.

This class provides the best way for an application developer to get programmatic access to what is happening on a queue. This may be simply to receive notifications of updates, or to serve as the basis for building a view system which is independent of that available with GDA.

gqmv-tabular-view-manager

This class serves as the intermediary between a gqm-queue and a tabular view. It implements the methods gqs-update-view-per-removal, gqs-update-view-per-addition, and gqs-update-view-per-attribute, which are required for all view managers.

The gqmv-tabular-view-manager also maintains the order of the queue entries as they appear on the tabular view. The attributes on this class serve that purpose.

This class is documented for reference only and should generally not be subclassed, nor should it be instantiated outside of the normal operation of the GQMV tabular view system.

Class Inheritance Path

gqmv-tabular-view-manager, gqsv-tabular-view-manager, gqs-view-manager, object, item

Attributes

Attribute	Description
gqsv-key-for-column-to-sort-initially	On creation, this attribute takes the value of the gqsv-root-specification attribute of the same name. This attribute is read only.
Allowable values:	Any symbol
Default value:	UNSPECIFIED
gqsv-automatically-resort-new-items	On creation, this attribute takes the value of the gqsv-root-specification attribute of the same name. This attribute is read only.
Allowable values:	Any truth-value
Default value:	false

Attribute	Description
gqsv-automatically-resort-attribute-changes	On creation, this attribute takes the value of the gqsv-root-specification attribute of the same name. This attribute is read only.
Allowable values:	Any truth-value
Default value:	false
gqsv-sorting-order	This value is automatically set by the gqsv-sort-order-button associated with the view. Care must be taken if operating programmatically on this attribute to keep this value synchronized with the gqsv-icon-state on that class.
Allowable values:	ASCENDING, DESCENDING
Default value:	ASCENDING
gqsv-scroll-to-new-items	On creation, this attribute takes the value of the gqsv-root-specification attribute of the same name. This attribute is read only.
Allowable values:	Any truth-value
Default value:	false
gqsv-update-latency	This attribute introduces a lag between events occurring on the queue and those events being reflected on the view. This is done to prevent the system from locking up trying to serve the UI in response to changes on the queue happening very rapidly. If you expect your queues to have very high throughput and you find that the system performs well with no view of the queue showing, but poorly when users are viewing the queues, you should set this value to 1.0 or higher (version 4.0 Rev. 0 and 4.0 Rev. 1 set this value to 1.0).
Allowable values:	Any float
Default value:	1.0

Extensible Methods

gqs-view-manager

This is an abstract (non-instantiable) class that provides a conduit for interaction between a queue and its views. If you want to create a new style of view, you must define a subclass of this class, and implement the three methods that define the behavior of your view in response to removal events, item addition events, and attribute change events. The signatures of these methods are:

gqs-update-view-per-removal
(*Manager*: class gqs-view-manager, *Queue*: class gqm-queue, *Items*: class item-list, *Client*: class object).

This method is called when one or more items (given in *Items*) have been removed from the queue.

gqs-update-view-per-addition
(*Manager*: class gqs-view-manager, *Queue*: class gqm-queue, *NewItems*: class item-list, *Win*: class g2-window).

This method is called when one or more items (given in *NewItems*) have been added to the queue.

gqs-update-view-per-attribute
(*Manager*: class gqs-view-manager, *Queue*: class gqm-queue, *Item*: class item, *Attribute*: symbol).

This method is called when an item contained in the queue receives a new attribute value. Normally, this requires that the view manager be monitoring for attribute changes. However, for two attributes on gda-alarm-entry, attribute updates occur even with no attributes being monitored. Specifically, these are the acknowledged attribute and a "virtual attribute," gda-alarm-status. See "gda-alarm-entry" on page 204 for more information.

In these methods, the "class gqs-view-manager" is replaced by the subclass name you create.

In addition, there is a method for deletion, but since removal implies deletion for any gqm-queue (see "gqs-remove-items" on page 198), the deletion method is redundant.

GDA uses a subclass of this view manager, gqmv-tabular-view-manager, so you do not have to create such a subclass to work with the standard queue views. However, the ability to subclass gqs-view-manager can provide very powerful functionality.

For example, suppose you were interested in providing a visual indication of the alarms generated by GDA from another system, such as the native UI of a DCS. Although it would be possible to construct such a system by subclassing entries or writing callbacks on the queue, using a custom view manager provides the cleanest, most modular, and most object-oriented approach to solving this problem. You simply create a subclass of gqs-view-manager, and then write the

three methods to contain the code to communicate with your remote UI (perhaps through JavaLink or ActiveXLink). That code can access the entries programmatically (*Item* argument or the contents of the *ItemList* argument). To activate the system, create an instance of your class and then subscribe it as a queue listener using the API "gqs-register-view" on page 291.

A registered view manager could also serve as a programmatic user. That is, a subclass of view manager could listen to a Queue, and take automated actions in lieu of an operator or end user.

Class Inheritance Path

gqs-view-manager, object, item

Attributes

none

Extensible Methods

"gqs-view-manager::gqs-update-view-per-addition" on page 371

"gqs-view-manager::gqs-update-view-per-attribute" on page 372

"gqs-view-manager::gqs-update-view-per-delete" on page 373

"gqs-view-manager::gqs-update-view-per-removal" on page 374

Relations

gqmv-view-manager-of-window

gqs-view-manager-for-queue

View Manager Procedures

This section describes procedures that are used to subscribe and unsubscribe a view manager to a queue.

gqs-deregister-view

Synopsis

gqs-deregister-view
(*Queue*: gqs-queue, *ViewManager*: gqs-view-manager, *Client*: object)

Argument	Description
Queue	The queue currently associated with the view manager.
ViewManager	The manager for the view.
Client	The client for this call.

Description

Use this procedure to disassociate a view and its underlying queue. See also "gqs-register-view" on page 291.

gqs-register-view

Synopsis

gqs-register-view
(*Queue*: gqs-queue, *ViewManager*: gqs-view-manager, *Client*: object)

Argument	Description
Queue	The queue to be associated with the given view manager.
ViewManager	Any view manager.
Client	The client for this call.

Description

Use this procedure to associate a view manager and a queue. A queue provides updates to each registered view manager. See "gqs-view-manager" on page 287 for more details on the relationship between queues and view managers. To reverse the effect of this call, use "gqs-deregister-view" on page 290.

gqsv-close-tabular-view

Synopsis

gqsv-close-tabular-view
(*Manager*: gqsv-tabular-view-manager, *Win*: g2-window)

Argument	Description
Manager	The manager for the view.
Win	The client for this call.

Description

This procedure hides the workspace of the view associated with *Manager* and releases the host dialog, if any, then deletes the view and its manager, and does related cleanup. The underlying queue is not affected by this procedure. See also "gqsv-delete-view" on page 293.

gqsv-delete-view

Synopsis

gqsv-delete-view
(*Manager*: gqsv-tabular-view-manager, *Client*: object)

Argument	Description
Manager	The manager for the view.
Client	The client for this call.

Description

This procedure deletes the view and its manager, and does related cleanup. The underlying queue is not affected by this procedure. See also "gqsv-close-tabular-view" on page 292.

Queue View Definitions

The section lists the API objects that relate to queue views. The queue system has a clear separation of the data component (discussed primarily in the earlier sections of this chapter) and the UI component. The views are the UI mechanism provided with this product. The views are configurable, as discussed in earlier sections, but to configure and customize them, it is useful to understand how they work.

Inside the View Template

The table portion of the tabular view is a GXL spreadsheet, described in detail in the *GXL User's Guide*. If you are modifying the GDA queues, you should understand how the spreadsheets are defined and what the spreadsheet views are. The tabular views presented in the GQMV module and the GDA product follow these concepts. As with any GXL spreadsheet, the tabular view is defined using a "specification." Ultimately, any UI that modifies the appearance of the table in GDA is going to be editing the specification for the view table. For simple edits to the spreadsheet definition (e.g., lines per row, font size), a dialog is presented to edit these options on the spreadsheet definitions. For more complex edits, the user can select the configure tabular view menu choice or the layout button. Either presents a sample table, which can be edited, but ultimately the changes are saved in the spreadsheet specification. The spreadsheet specification is based on a special queue class, gqsv-root-specification, but otherwise uses the specification blocks as defined for GXL.

The second part to defining a view template is the definition of the workspace upon which the tabular view sits. The GQMV module creates specific instances of a view by cloning this master workspace and putting a table (as defined in the specification above) on that workspace. Where it finds that master workspace depends on the attribute gqsv-master-dialog on the View Template object. If that attribute contains a uil-dialog, then that dialog is reserved and used as any other dialog with UIL (see the GUIDE manuals if you intend to use this feature). If an object is named, then the subworkspace of that object is cloned and displayed as the view. If the value of this attribute is UNSPECIFIED, as it is for any built-in view templates within GDA, the system finds a connection post on the subworkspace of the template. That connection post must then have a subworkspace, and that subworkspace is cloned and used to display the view

Detail Templates

In addition to the templates of class gqmv-tabular-view-template, two classes provide what is referred to as "detail views:" gqmv-composition-view-template and gqmv-detail-view-template. These templates are much simpler in their operation as they simply clone their own subworkspace, and display that as instances of their views.

gqmv-button-setting

The gqmv-button-setting is what the buttons on the gqmv-tabular-view workspace use to determine the location of the subviews they launch. See gqmv-view-entry-details-button. See the *GDA User's Guide* for more information.

This class inherits from both gqsv-workspace-location and gfr-module-setting. To access a module setting for writing, be sure to use the API "gdl-get-active-setting" on page 26.

Class Inheritance Path

gqmv-button-setting, gfr-module-setting, gqsv-workspace-location, object, item

Attributes

Attribute	Description
x-scale	See "gqsv-workspace-location" on page 321.
Allowable values:	Inherited
Default value:	1.0
y-scale	See "gqsv-workspace-location" on page 321.
Allowable values:	Inherited
Default value:	1.0
window-x-location	See "gqsv-workspace-location" on page 321.
Allowable values:	Inherited
Default value:	CENTER
window-y-location	See "gqsv-workspace-location" on page 321.
Allowable values:	Inherited
Default value:	CENTER

Attribute	Description
workspace-x-location	See "gqsv-workspace-location" on page 321.
Allowable values:	Inherited
Default value:	CENTER
workspace-y-location	See "gqsv-workspace-location" on page 321.
Allowable values:	Inherited
Default value:	CENTER

gqmv-composition-box

This class has no attributes or extensible methods. An instance of gqmv-composition-box must appear on the subworkpace of a gqmv-composition-view-template for it to work properly.

Class Inheritance Path

gqmv-composition-box, gqmv-view, _gqmv-vertical-scrollable-area, _gqmv-scrollable-area, gqmv-message, message, item

Attributes

none

Relations

the-gqmv-detail-editor-of

gqmv-composition-view-template

This class is very similar to gqmv-detail-view-template, except its purpose is to contain the workspace that enables users to add comments to an entry. To launch a comment session, use the API "gqmv-launch-comment-editor-from-view" on page 326.

The class has no public attributes or methods. To use a custom instance of this class, that instance must have a subworkspace and that subworkpace must have on it a gqmv-composition-box. For all existing queues, the built-in instance gqmv-default-composition-template is used.

Class Inheritance Path

gqmv-composition-view-template, gqmv-template, object, item

Attributes

none

gqmv-count-display

The existence of a count display is what distinguishes gqmv-tabular-view-template from its parent class. Typically, two instances of this class appear on the workspace of a tabular view. The system finds these instances, if they exist, and updates them appropriately according to their gqmv-count-id.

Class Inheritance Path

gqmv-count-display, gqmv-message, message, item

Attributes

Attribute	Description
gqmv-count-id	This should either be queue-count or view-count depending on whether the display is showing the number of entries in the queue, or the number of entries showing in the view.
Allowable values:	Any text
Default value:	""

gqmv-detail-view-template

This is the object that holds template workspaces for the detail views. It is analogous to the gqmv-tabular-view-template for the primary queue views. To launch a detail view of an entry, use the API "gqmv-launch-detail-view" on page 327.

The class has no public attributes or methods. To use a custom instance of this class, that instance must have a subworkspace and that subworkpace must have on it a gqmv-view.

There are two built-in instances of this class used by GDA. One is for the alarm queue details, and the other is used by all other queue types.

Class Inheritance Path

gqmv-detail-view-template, gqmv-template, object, item

Attributes

none

gqmv-tabular-view-template

This class extends gqsv-tabular-view-template by recognizing the existence of count displays on views (see "gqmv-count-display" on page 299). The class also simplifies the configuration of a tabular view by adding column height and font size to the configuration dialog.

GDA provides four default instances of this class, one for each of the built-in queues. You can create new instances of this class by cloning the built-in instances or by programmatically creating, transferring, and making permanent a new instance of this class. If the template object does not have a subworkspace, you can create a new subworkspace and basic template by selecting the configure-tabular-view menu choice. When you do, you see a facsimile of the view, where you set the properties of the view by editing the tables of the columns, column headers, and border. See "gqsv-column" on page 306, "gqsv-column-header" on page 310, and "gqsv-view-configuration" on page 319 for more information.

Use the API procedure "gqs-create-view" on page 331 to launch a new tabular view programmatically.

Class Inheritance Path

gqmv-tabular-view-template, gqsv-tabular-view-template, gqmv-template, object, item

Attributes

Attribute	Description
gqmv-font-size	This attribute provides a convenient place to edit and store the font size of the columns. This corresponds to the gqmv-column attribute font-size, or the gxl-font-size in the specifications. This value cannot be edited directly. It must be configured by the user through the configuration dialog (the configure menu choice on the tabular view).
Allowable values:	Any symbol
Default value:	SMALL
gqmv-lines-per-row	This attribute provides a shortcut to laying out the row height in a tabular view edit session. This value cannot be edited directly. It must be configured by the user through the configuration dialog (the configure menu choice on the tabular view).

Attribute	Description
Allowable values:	Any integer
Default value:	1
gqsv-view-manager-class	See "gqsv-tabular-view-template" on page 317
Allowable values:	Inherited
Default value:	GQSV-TABULAR-VIEW-MANAGER
gqsv-master-dialog	See "gqsv-tabular-view-template" on page 317
Allowable values:	Inherited
Default value:	UNSPECIFIED
gqsv-initial-view-location	See "gqsv-tabular-view-template" on page 317
Allowable values:	Inherited
Default value:	GQSV-WORKSPACE-LOCATION
gqsv-monitor-deletion-events-on-visible-items	See "gqsv-tabular-view-template" on page 317
Allowable values:	Inherited
Default value:	true

Extensible Methods

"gqmv-tabular-view-template::gqs-create-view" on page 367

gqmv-view

This class has no attributes or extensible methods. An instance of gqmv-view must appear on the subworkpace of a gqmv-detail-view-template for it to work properly.

Class Inheritance Path

gqmv-view, _gqmv-vertical-scrollable-area, _gqmv-scrollable-area, gqmv-message, message, item

Attributes

none

Relations

the-gqmv-detail-view-of

the-gqmv-detail-editor-of

gqs-queue-access-table

This class is used to specify which template is to be used to create a view for a given user or window. The access table consists of three parallel arrays, giving a user name or window class, a user mode, and a template name, respectively. Given a window, the access table is searched from the first element of the array to the last, stopping when user name or window class and user mode match the properties of the given g2-window. The matching procedure is implemented by gqs-get-view-template.

The easiest way to specify a queue access table is to clone one from the GQS palette, and then select the configure access table menu choice. Fill in the table specifying a user name or window class in the first column, a G2 user mode in the second column, and the desired template name in the third column. "Any" is acceptable in the g2-user-mode column. Add as many rows as required in your table, using the GXL add row control button.

For example, suppose you want to associate Template-1 with User-1 regardless of user mode, Template-2 with any user in administrator mode, and Template-3 for any user in any other mode. Then, the three arrays or columns would be:

gqs-user-name-or-window-class = (User-1, g2-window, g2-window)

gqs-user-mode = (ANY, ADMINISTRATOR, ANY)

gqs-view-template = (TEMPLATE-1, TEMPLATE-2, TEMPLATE-3)

Once the access table is created, give it a name, then set the gqs-view-template-or-access-table attribute of a gqs-queue to the name of your access table.

Class Inheritance Path

gqs-queue-access-table, object, item

Attributes

Attribute	Description
gqs-user-name-or-window-class	A list of user names or G2 window classes.
Allowable values:	Any symbol-array
Default value:	SYMBOL-ARRAY
gqs-user-mode	A list of user modes corresponding to the list of user names/window classes. The symbol ANY can be used if the mode is not part of the selection criteria.
Allowable values:	Any symbol-array
Default value:	SYMBOL-ARRAY
gqs-view-template	A list of template names corresponding to the user name/window class and user mode.
Allowable values:	Any symbol-array
Default value:	SYMBOL-ARRAY

gqsv-column

The purpose of this class is to enable users to configure the tabular view. It does not have a persistent life of its own once the view template is closed and should not be referenced as an API. None of the attributes of the class should be accessed programmatically.

This class lets you configure the properties of the cells in a column on a queue view. An instance of this class appears below each column header (see "gqsv-column-header" on page 310) in the view when you interactively configure a gqsv-tabular-view-template. You set the desired properties of the cells by editing the attributes of the gqsv-column, through its table. Also see "gqsv-view-configuration" on page 319.

Class Inheritance Path

gqsv-column, gqsv-column-or-header, gqsv-configuration-message, message, item

Attributes

Attribute	Description
number-of-visible-rows	The number of rows to display in the view. This property is common to all columns. That is, editing this property in one of the columns on a table also changes it for all of the other columns.
Allowable values:	Any positive integer
Default value:	5
cell-height	The cell height, in number of pixels.
Allowable values:	Any positive integer
Default value:	28
cell-type	The type of cell used in the column. Specify the most specific cell type consistent with the values to be displayed. For any attribute displayed on the built-in queue views, it is probably not appropriate to change this value from the default.

Attribute	Description
Allowable values:	VALUE-CELL, INTEGER-CELL, FLOAT-CELL, QUANTITY-CELL, TEXT-CELL, SYMBOL-CELL, TRUTH-VALUE-CELL
Default value:	VALUE-CELL
initialization-data	A value passed to the initialization procedure for the cell. Normally left as UNSPECIFIED.
Allowable values:	Any value
Default value:	UNSPECIFIED
cells-are-selectable	A truth-value indicating if clicking on the cell should select (highlight) the cell. For the GDA queues, this is generally left as true, otherwise the views will not behave as intended.
Allowable values:	Any truth-value
Default value:	true
cells-are-editable	A truth-value indicating if the user should be able to interactively edit the contents of the cell. For the GDA queues, this value is set to false. The GDA queue views are designed to be display only. The data behind the views represents information found in the source objects.
Allowable values:	Any truth-value
Default value:	false

Attribute	Description
validation-procedure	The name of an optional user-written procedure that validates user input, if the cells in the column are editable. For the reason described above, this should not normally be used. However, if this procedure is used, the signature of your validation method must be: (Sheet: class gxl-spreadsheet, Row: integer, Col: integer, NewValue: value, Win: class g2-window) = (ValueOrErrorText: value, Accepted: truth-value). Note that basic type checking is done automatically based on the cell type, and does not require a validation procedure. See the *GXL User's Guide* for more information.
Allowable values:	Any symbol
Default value:	UNSPECIFIED
callback-procedure	The name of an optional user-written procedure that is called when the cell receives a new value. This callback might be used if you want view-specific behavior to be executed when a view is updated. Note that the gqsv-column-header contains an attribute for updating cell colors. If you have a reason to insert a procedure here, the signature of your validation method must be: (Sheet: class gxl-spreadsheet, Row: integer, Col: integer, NewValue: value, Win: class g2-window). See the *GXL User's Guide* for more information.
Allowable values:	Any symbol
Default value:	GQSV-SET-ATTRIBUTE-CALLBACK
selection-callback-procedure	The name of an optional user-written procedure that is called when any cell in the column is selected. The signature of the selection callback you provide is: (Sheet: class gxl-spreadsheet, View: class gxl-spreadsheet-view, FirstRow: integer, FirstCol: integer, NumberOfRows: integer, NumberOfCols: integer, Win: class g2-window). See the *GXL User's Guide* for more information.
Allowable values:	Any symbol

Attribute	Description
Default value:	UNSPECIFIED
float-format	Edit the subtable of this attribute to change the formatting of floating point numbers shown in this column. See the *GXL User's Guide* for more information.
Allowable values:	Any gxl-float-formatter
Default value:	GXL-FLOAT-FORMATTER
font-size	See "gqsv-column-or-header" on page 314.
Allowable values:	SMALL, LARGE, EXTRA-LARGE
Default value:	SMALL
default-background-color	See "gqsv-column-or-header" on page 314.
Allowable values:	Inherited
Default value:	WHITE
default-text-color	See "gqsv-column-or-header" on page 314.
Allowable values:	Inherited
Default value:	BLACK
default-border-color	See "gqsv-column-or-header" on page 314.
Allowable values:	Inherited
Default value:	BLACK

gqsv-column-header

The purpose of this class is to enable users to configure the tabular view. It does not have a persistent life of its own once the view template is closed and should not be referenced as an API. None of the attributes of the class should be accessed programmatically.

This class lets you configure the properties of a column on a queue view. An instance of this class appears at the top of each column of the view when you interactively configure a gqsv-tabular-view-template. You set the desired properties of each column by editing the attributes of the gqsv-column-header, through its table. Also see "gqsv-view-configuration" on page 319 and "gqsv-column" on page 306.

Class Inheritance Path

gqsv-column-header, gqsv-column-or-header, gqsv-configuration-message, message, item

Attributes

Attribute	Description
visible-label	The text that appears at the top of the column, which usually represents the property shown in the column.
Allowable values:	Any text
Default value:	"Any label"
attribute-or-key	The name of the attribute that is displayed in this column, or a symbolic key for the column if it is not directly displaying an attribute. In the latter case, the key given here is passed to the key-value-conversion-procedure.
Allowable values:	Any symbol
Default value:	UNSPECIFIED

Attribute	Description
key-value-conversion-procedure	The name of the procedure that provides values for the entries in the column. If the attribute-or-key names an attribute, the default for this attribute (gqsv-get-attribute-value) simply retrieves the value of the named attribute. If you want to specially format the values appearing in this column (for example, representing a float as a timestamp), or if you want to display a value that is not an attribute of the underlying item, you must provide your own key-value-conversion-procedure. The signature of your procedure must be: (Item: class item, Key: symbol) = (value), where Key is the attribute-or-key for this column, the item an item in the queue, and the return argument is the value to be displayed in the view.
Allowable values:	Any symbol
Default value:	GQSV-GET-ATTRIBUTE-VALUE
dynamic-color-formatter	The name of an optional procedure that provides color information for cells in this column. The signature of this procedure is: (Item: class item, Key: symbol) = (symbol, symbol, symbol), where Item is the item in the queue, Key is the attribute-or-key for this column, and the return arguments are the desired background, text, and border color for the cell.
Allowable values:	Any symbol
Default value:	UNSPECIFIED
width	The width, in pixels, of this column.
Allowable values:	Any integer
Default value:	120
allow-click-to-sort	Controls whether clicking the column header causes the view to sort on the values in the column. If sorting of this column is not allowed, set this attribute to false.
Allowable values:	Any truth-value

Attribute	Description
Default value:	true
monitor-this-attribute	Setting this value to true indicates to the view that it should continuously monitor the attribute displayed in this column for changes. For the built-in views and the default attributes, this value is always set to false for two reasons. **1** First, most of the attributes are set only on entry creation, and can never change for a given entry. For example, a time stamp is the creation time of the entry and can never be expected to change. **2** Second, the attributes that are expected to change are designed to update the views without the overhead of attribute monitoring. There are two such attributes on alarm entries that can be expected to change. See "gda-alarm-entry" on page 204 and the attributes acknowledged and gda-alarm-status for more information. If you created new columns, you should leave this attribute as true (the default) if you expect that value to be changing dynamically, and you want the dynamic changes to be updated in the view.
Allowable values:	Any truth-value
Default value:	true
font-size	See "gqsv-column-or-header" on page 314.
Allowable values:	SMALL, LARGE, EXTRA-LARGE
Default value:	LARGE
default-background-color	See "gqsv-column-or-header" on page 314.
Allowable values:	Inherited

Attribute	Description
Default value:	WHITE
default-text-color	See "gqsv-column-or-header" on page 314.
Allowable values:	Inherited
Default value:	BLACK
default-border-color	See "gqsv-column-or-header" on page 314.
Allowable values:	Inherited
Default value:	BLACK

gqsv-column-or-header

This is a non-instantiable class that is a parent for gqsv-column-header and gqsv-column, and contains attributes common to both classes, both used in view template configuration.

Class Inheritance Path

gqsv-column-or-header, gqsv-configuration-message, message, item

Attributes

Attribute	Description
font-size	Controls the font size in the view.
Allowable values:	SMALL, LARGE, EXTRA-LARGE
Default value:	SMALL
default-background-color	Controls the background color of the cells in the view, if the background color has not otherwise been set.
Allowable values:	Any symbol
Default value:	WHITE
default-text-color	Controls the text color of the cells in the view, if the text color has not otherwise been set.
Allowable values:	Any symbol
Default value:	BLACK
default-border-color	Controls the border color of the cells in the view, if the border color has not otherwise been set.
Allowable values:	Any symbol
Default value:	BLACK

gqsv-root-specification

Instances of this object reside on the subworkspace of any configured gqmv-tabular-view-template, along with other gxl-specification-objects. Normally, you would not access these objects directly, instead using the graphical configuration tools when you chose Configure Tabular View on the gqmv-tabular-view object (see also "gqsv-column" on page 306, "gqsv-column-header" on page 310, and "gqsv-view-configuration" on page 319).

You can, however, access these objects programmatically. See the *GXL User's Guide* for more information.

Class Inheritance Path

gqsv-root-specification, gxl-root-specification, gxl-specification-object, gfr-object-with-uuid, object, gfr-item-with-uuid, item

Attributes

Attribute	Description
gqsv-add-new-items-first-or-last	See "gqsv-view-configuration" on page 319 for a description of this attribute.
Allowable values:	Any symbol
Default value:	LAST
gqsv-scroll-to-new-items	See "gqsv-view-configuration" on page 319 for a description of this attribute.
Allowable values:	Any truth-value
Default value:	false
gqsv-automatically-resort-new-items	See "gqsv-view-configuration" on page 319 for a description of this attribute.
Allowable values:	Any truth-value
Default value:	false

Attribute	Description
gqsv-automatically-resort-attribute-changes	Indicates whether or not an attribute change should cause the view to be re-sorted. For example, if a view is sorting on the acknowledged attribute, you may want entries that are acknowledged to be sent to the bottom of the view immediately upon acknowledgement. See also "gqsv-view-configuration" on page 319 for a description of this attribute.
Allowable values:	Any truth-value
Default value:	false
gqsv-key-for-column-to-sort-initially	Configured through the gqmv-tabular-view-template configuration dialog. See also "gqsv-view-configuration" on page 319 for a description of this attribute.
Allowable values:	Any symbol
Default value:	UNSPECIFIED
gqsv-initial-sorting-order	Configured through the gqmv-tabular-view-template configuration dialog. See also "gqsv-view-configuration" on page 319 for a description of this attribute.
Allowable values:	ASCENDING, DESCENDING
Default value:	ASCENDING

gqsv-tabular-view-template

This object defines the layout and behavior of a queue view. This class should not be subclassed or extended within the context of GDA. Instead use the class "gqmv-tabular-view-template" on page 301.

Class Inheritance Path

gqsv-tabular-view-template, object, item

Attributes

Attribute	Description
gqsv-view-manager-class	The class of view manager associated with this view.
Allowable values:	Any symbol
Default value:	GQSV-TABULAR-VIEW-MANAGER
gqsv-master-dialog	If the view is to be created on a uil-dialog or any other special workspace, specify the dialog ID here, or specify the name of the superior item of the target workspace. When you launch the view, GQS either reserves the appropriate dialog and builds the view on the dialog, or clones the subworkspace of the named superior item, and builds the view on the cloned workspace.
Allowable values:	Any value
Default value:	UNSPECIFIED
gqsv-initial-view-location	The location in the host window, where the workspace of the view is to be shown. See "gqsv-workspace-location" on page 321 for details.
Allowable values:	Any gqsv-workspace-location
Default value:	GQSV-WORKSPACE-LOCATION

Attribute	Description
gqsv-monitor-deletion-events-on-visible-items	Controls if the view sets up monitors on the deletion of items visible in the view. • When this attribute is true, the view automatically updates if one of the displayed items is deleted. • If the underlying queue is already monitoring deletion events, this attribute should be false. Monitoring deletion events for visible items only is more efficient than monitoring all items in the queue, since only a small subset of items in the queue may be visible at any one time.
Allowable values:	Any truth-value
Default value:	true

gqsv-view-configuration

The purpose of this class is to enable users to configure the tabular view. It does not have a persistent life of its own once the view template is closed and should not be referenced as an API. None of the attributes of the class should be accessed programmatically.

This class lets you configure the properties of a queue view. This class is the frame around the cells in the view facsimile shown when you interactively configure a gqsv-tabular-view-template. You set the desired properties of the view by editing the attributes of the gqsv-view-configuration. Show its table by clicking on the edge of the view facsimile. Also, see "gqsv-column-header" on page 310 and "gqsv-column" on page 306.

Class Inheritance Path

gqsv-view-configuration, gqsv-configuration-message, message, item

Attributes

Attribute	Description
add-new-items-first-or-last	Determines whether new items are added to the top or bottom of the view. Normally, adding new items last (at the bottom of the view) is more efficient than adding items first (to the top of the view), because screen redraws are needed less frequently.
Allowable values:	Any symbol
Default value:	LAST
scroll-to-display-new-items	Determines whether the queue view should automatically scroll to show any new items added to the view. For example, if this attribute is set to true, when items are added to the bottom of the view, the view scrolls to expose the last item in the view.
Allowable values:	Any truth-value
Default value:	false

Attribute	Description
automatically-resort-new-items	Determines if the view should be sorted according to the current sort column, whenever new items are added to the view. Note that there may be a substantial performance penalty for using this feature.
Allowable values:	Any truth-value
Default value:	false
automatically-resort-attribute-changes	Determines if the view should be sorted according to the current sort column whenever the associated attribute of any item in the queue receives a new value. This feature is only relevant if the queue is monitoring the attribute associated with the current sort column. Note that there may be a substantial performance penalty for using this feature.
Allowable values:	Any truth-value
Default value:	false
key-for-column-to-sort-initially	If the view is configured to sort or resort automatically, this attribute determines which column are sorted when the view is initially displayed. The column is specified by its attribute-or-key, given in the column-header. Note that the user can change the sort column interactively once the view is displayed by clicking on the column header.
Allowable values:	Any symbol
Default value:	UNSPECIFIED
initial-sorting-order	The order of the initial sorting of the view. Note that the user may be able to interactively change the sort order (see the *GDA User's Guide*).
Allowable values:	ASCENDING, DESCENDING
Default value:	ASCENDING

gqsv-workspace-location

This item provides a generic way to specify the location and scale of a workspace within a g2-window. The workspace is shown by placing a focal point on the workspace (given by workspace-x-location and workspace-y-location) at the given point in the window (given by window-x-location and window-y-location). For an application of this class, see "gqsv-tabular-view-template" on page 317.

Class Inheritance Path

gqsv-workspace-location, object, item

Attributes

Attribute	Description
x-scale	The horizontal scale of the workspace, where 0.5 is half-size, 1.0 is full size, 2.0 is twice normal scale, etc.
Allowable values:	Any positive float
Default value:	1.0
y-scale	The vertical scale of the workspace, where 0.5 is half-size, 1.0 is full size, 2.0 is twice normal scale, etc.
Allowable values:	Any positive float
Default value:	1.0
window-x-location	This value can be either the symbol TOP, CENTER, or BOTTOM, or an integer indicating a horizontal coordinate in the window coordinates.
Allowable values:	Any value
Default value:	CENTER
window-y-location	This value can be either the symbol TOP, CENTER, or BOTTOM, or an integer indicating a vertical coordinate in the window coordinates.

Attribute	Description
Allowable values:	Any value
Default value:	CENTER
workspace-x-location	This value can be either the symbol TOP, CENTER, or BOTTOM, or an integer indicating a horizontal coordinate in workspace coordinates.
Allowable values:	Any value
Default value:	CENTER
workspace-y-location	This value can be either the symbol TOP, CENTER, or BOTTOM, or an integer indicating a vertical coordinate in workspace coordinates.
Allowable values:	Any value
Default value:	CENTER

Queue View Procedures

These procedures apply to queue views. The three important procedures are gqs-create-view, gqmv-launch-detail-view, and gqmv-launch-comment-editor-from-view. These are the procedures to launch a view of a queue, a detail view of an entry, and a comment editor session for a detail view, respectively.

The remaining procedures enable access to some of the functions used in working with detail views. These procedures may not be supported in future versions of the API.

gqm-time-to-text

Synopsis

gqm-time-to-text
(*UserTime*: float, *Format*: symbol)
–> *Text*

Argument	Description
UserTime	The G2 time.
Format	A symbol that can be one of: DATE, TIME, DATETIME, FILETIME (a format that is suitable for use in filenames requiring a date-time segment).

Return Value	Description
Text	The formatted time.

Description

Returns a formatted time given a time and format. These formats can be edited in the gqm-general-setting preference (configured via Preferences > Queues > General Settings). This procedure is callable if you need to obtain a formatted text string from a G2 time and you want it to be configured as in the gqm-general-setting.

gqmv-detail-array-to-text

Synopsis

gqmv-detail-array-to-text
(*TextLines*: text-array, *Client*: g2-window)
–> *Text*

Argument	Description
TextLines	A text array containing lines of text in each element of the array.
Client	The client for this call.

Return Value	Description
Text	The resulting text.

Description

This API procedure returns the contents of a text array as a single text, but with the original elements numbered. Its purpose is to convert data contained in text arrays (such as comments) into a single text for display.

gqmv-launch-comment-editor-from-view

Synopsis

gqmv-launch-comment-editor-from-view
(*Entry*: gqm-entry, *Template*: gqmv-composition-view-template,
SuperiorView: gqmv-view, *ViewLocation*: gqsv-workspace-location,
ShowIndices: truth-value, *Win*: g2-window)

Argument	Description
Entry	The entry whose gqm-comments field is to be edited.
Template	A template specifying the subview layout for editing the comments of *Entry*.
SuperiorView	A gqmv-view that is displaying the details of *Entry*. This can be created via the call gqmv-launch-detail-view (Entry, DetailTemplate (A template specifying the layout of the Superior View), Win). When the comments are entered via the comment editor, all comments are displayed in the *SuperiorView*.
ViewLocation	A gqsv-workspace-location object that specifies where on the window the comment editor should appear. The default location can be obtained via the call gfr-get-active-setting (the symbol GQMV-BUTTON-SETTING, Win);
ShowIndices	Specifies whether the comments are numbered when inserted back into *SuperiorView*.
Win	The window on which the editor is to be displayed.

Description

Use this procedure to launch a comment editor session. It is necessary to have the entry already displayed in a detail view, and gqmv-view from that detail view must be included in this API.

gqmv-launch-detail-view

Synopsis

gqmv-launch-detail-view
(*Entry*: gqm-entry, *Template*: gqmv-detail-view-template, *Win*: g2-window)
–> *gqmv-View*

Argument	Description
Entry	The entry whose details are to be displayed.
Template	A template, appropriate to the class of *Entry*, for displaying *Entry*'s information.
Win	The g2-window on which to display the view.

Return Value	Description
gqmv-View	The view created by this procedure. By default, a gqmv-view are deleted when dismissed by the user.

Description

If a detail view of *Entry* already exists, then that view is returned. If not, a new view is created, shown on *Win* and the newly created view is returned. The view is displayed using the default gqmv-button-setting (which is configured via the main menu).

gqmv-show-workspace

Synopsis

gqmv-show-workspace
(*Workspace*: kb-workspace, *Location*: gqsv-workspace-location,
Client: g2-window)

Argument	Description
Workspace	The workspace to show.
Location	The location to display the workspace.
Client	The window.

Description

Shows a workspace at a designated location on a window

gqmv-text-array-to-line-array

Synopsis

gqmv-text-array-to-line-array
(*SourceArray*: text-array, *TextLines*: text-array, *ScrollAreaWidth*: integer, *FontSize*: symbol, *ShowIndex*: truth-value, *Client*: g2-window)

Argument	Description
SourceArray	Text array to parse.
TextLines	The array of parsed text lines.
ScrollAreaWidth	The view width.
FontSize	The G2 font size.
ShowIndex	If true, shows the line index starting at 1.
Client	The window.

Description

Parses a text array to an array of text. Each parsed text element is one line on the display.

gqmv-text-to-line-array

Synopsis

gqmv-text-to-line-array
(*SourceText*: text, *TextLines*: text-array, *ScrollAreaWidth*: integer, *FontSize*: symbol, *Client*: g2-window)

Argument	Description
SourceText	Text to parse.
TextLines	The array for the parse lines.
ScrollAreaWidth	The width of the scroll area/view.
FontSize	The G2 font size.
Client	The window.

Description

Parses the source text into an array. Each text in the array contains one display line of text.

gqs-create-view

Synopsis

gqs-create-view
(*Template*: item, *Queue*: gqs-queue, *Client*: object)
–> *gqs-View-Manager*

Argument	Description
Template	Any item that provides the specification of a view by implementing this method.
Queue	The queue to which the view is to be linked.
Client	The client where the view is to be shown. Note that for gqmv-tabular-view-template, this argument must be a g2-window. For arbitrary user extensions of this method, however, this may be a ui-client-item.

Return Value	Description
gqs-View-Manager	The manager for the view. Typically this is a gqmv-tabular-view-manager, but it could be any subclass of gqs-view-manager if that has been implemented.

Description

This procedure enables you to create a view programmatically, given a template and a queue. It has the same effect as launching a view from the create-view menu choice, except that the menu choice can use a view access table to select the proper template.

gqs-get-view-template

Synopsis

gqs-get-view-template
(*Table*: gqs-queue-access-table, *Win*: g2-window)
–> *Symbol*

Argument	Description
Table	Any queue access table.
Win	The window where the queue view is to be launched.

Return Value	Description
Symbol	The name of the template for the view to be used on this window.

Description

This procedure interrogates a queue access table and determines the proper view template for the given window. The search in the access table proceeds in the following order.

1. First, there is an attempt to match by user name.
2. If the user name does not exist or no match is found, then the search is conducted by window class.

In each case, a match also requires matching the user mode of the window. See "gqs-queue-access-table" on page 304 for more details.

Button Definitions

This section describes the button classes used on the views and applies to both the buttons used on the tabular view templates and those used on the detail views.

This section does not include descriptions of the actions performed when the buttons are used. For those descriptions, see the *GDA User's Guide*. Instead, this section attempts to provide the information you need if you intend to subclass and customize buttons.

gda-clear-alarm-entries-button

This button is subclassed so it can acknowledge every entry before deleting it. It also has a different criterion for when to prompt for confirm; it refers to the configuration on the gda-alarm-queue with which it is associated. This button also logs the clear action if logging is enabled.

Class Inheritance Path

gda-clear-alarm-entries-button, gqmv-clear-entries-button, gqsv-toolbar-button, gxl-toolbar-button, gxl-control-button, object, item

Attributes

Attribute	Description
gxl-help-text	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQMV-CLEAR-ENTRIES-HELP
gxl-help-resource	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQMV-TEXT-RESOURCES
gxl-button-state	See "gqsv-toolbar-button" on page 345.
Allowable values:	ENABLED, DISABLED, RELEASED
Default value:	RELEASED

Extensible Methods

These methods are described in the *GXL User's Guide*.

gxl-control-button::gxl-reflect-selection-state-in-button

gxl-control-button::gxl-set-button-state

gxl-control-button::gxl-perform-button-function

gda-remove-alarm-entries-button

The remove entry button for view of the alarm queue has several extensions beyond the basic behavior. Alarm entries that have not been acknowledged cannot be removed from the queue (see "gda-alarm-queue::gqs-remove-items" on page 352). This button is not enabled if entries are unacknowledged and gda-acknowledge-selected-entries is false.

If gda-acknowledge-selected-entries is true, then this button acknowledges all the selected entries before removing the entries, as per gqsv-remove-item-button.

Class Inheritance Path

gda-remove-alarm-entries-button, gqsv-remove-item-button, gqsv-multiple-row-button, gqsv-toolbar-button, gxl-toolbar-button, gxl-control-button, object, item

Attributes

Attribute	Description
gda-acknowledge-selected-entries	A truth value indicating whether or not the remove button should acknowledge any selected entries before removing them. If this attribute is false, this button is not active when selecting unacknowledged alarm entries. See "gda-alarm-entry" on page 204.
Allowable values:	Any truth-value
Default value:	false
gxl-help-text	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQSV-REMOVE-ITEM-HELP
gxl-help-resource	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQSV-TEXT-RESOURCES

Attribute	Description
gxl-button-state	See “gqsv-toolbar-button” on page 345.
Allowable values:	ENABLED, DISABLED, RELEASED
Default value:	RELEASED

Extensible Methods

gxl-control-button::gxl-reflect-selection-state-in-button

gxl-control-button::gxl-set-button-state

gxl-control-button::gxl-perform-button-function

gqmv-clear-entries-button

Removes all viewed entries from the queue.

Class Inheritance Path

gqmv-clear-entries-button, gqsv-toolbar-button, gxl-toolbar-button, gxl-control-button, object, item

Attributes

Attribute	Description
gxl-help-text	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQMV-CLEAR-ENTRIES-HELP
gxl-help-resource	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQMV-TEXT-RESOURCES
gxl-button-state	See "gqsv-toolbar-button" on page 345.
Allowable values:	ENABLED, DISABLED, RELEASED
Default value:	RELEASED

Extensible Methods

gxl-control-button::gxl-reflect-selection-state-in-button

gxl-control-button::gxl-set-button-state

gxl-control-button::gxl-perform-button-function

gqmv-go-to-source-button

The gqmv-go-to-source-button is different from its superior in that it is the gqm-source of the entry, not the entry itself to which the user typically wants to navigate. This button locates the source through the method the-gqm-source-of and then highlights it with a call to the procedure named in the gqsv-highlight-procedure of the button.

Class Inheritance Path

gqmv-go-to-source-button, gqsv-go-to-source-button, gqsv-single-row-button, gqsv-toolbar-button, gxl-toolbar-button, gxl-control-button, object, item

Attributes

Attribute	Description
gqsv-highlight-procedure	The name of a procedure that highlights the item navigated to by this button. The signature of the procedure must be: (Itm: class item, Win: class g2-window).
Allowable values:	Any symbol
Default value:	GQSV-HIGHLIGHT-ITEM
gxl-help-text	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQSV-GOTO-HELP
gxl-help-resource	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQSV-TEXT-RESOURCES
gxl-button-state	See "gqsv-toolbar-button" on page 345.

Attribute	Description
Allowable values:	ENABLED, DISABLED, RELEASED
Default value:	RELEASED

Extensible Methods

gxl-control-button::gxl-reflect-selection-state-in-button

gxl-control-button::gxl-set-button-state

gxl-control-button::gxl-perform-button-function

gqmv-save-selected-button

This button prompts the user for a filename, then saves the entry to that file using the gqm-save-entry API.

Class Inheritance Path

gqmv-save-selected-button, gqmv-toolbar-button, gqsv-single-row-button, gqsv-toolbar-button, gxl-toolbar-button, gxl-control-button, gqmv-file-button, gqmv-button, object, item

Attributes

Attribute	Description
gxl-help-text	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gxl-help-resource	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQMV-TEXT-RESOURCES
gxl-button-state	See "gqsv-toolbar-button" on page 345.
Allowable values:	ENABLED, DISABLED, RELEASED
Default value:	RELEASED
gqmv-filename-template	This text provides a template for generating the name of the file to which the selected entry is saved. The template is converted to a filename though a call to gqmv-get-filename-from-button.
Allowable values:	Inherited
Default value:	"save-selected-*.text"

Extensible Methods

gxl-control-button::gxl-reflect-selection-state-in-button

gxl-control-button::gxl-set-button-state

gxl-control-button::gxl-perform-button-function

gqsv-close-view-button

This button, which looks similar to the hide-workspace button, closes a queue view, deleting the view. It has the same effect as calling "gqsv-close-tabular-view" on page 292.

Class Inheritance Path

gqsv-close-view-button, gqsv-toolbar-button, gxl-toolbar-button, gxl-control-button, object, item

Attributes

Attribute	Description
gxl-help-text	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gxl-help-resource	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQSV-TEXT-RESOURCES
gxl-button-state	See "gqsv-toolbar-button" on page 345.
Allowable values:	ENABLED, DISABLED, RELEASED
Default value:	RELEASED

Extensible Methods

gxl-control-button::gxl-reflect-selection-state-in-button

gxl-control-button::gxl-set-button-state

gxl-control-button::gxl-perform-button-function

gqsv-multiple-row-button

This class can be used as a parent class for custom buttons on queue views. It defines the methods gxl-set-button-state and gxl-reflect-selection-state-in-button so that the button is active when any row or rows are selected in the view. If you subclass from this class, your icon must include the regions icon-highlight and icon-symbol. See "gqsv-single-row-button" on page 344 and "gqsv-toolbar-button" on page 345 for alternative parent classes.

Class Inheritance Path

gqsv-multiple-row-button, gqsv-toolbar-button, gxl-toolbar-button, gxl-control-button, object, item

Attributes

Attribute	Description
gxl-help-text	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gxl-help-resource	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQSV-TEXT-RESOURCES
gxl-button-state	See "gqsv-toolbar-button" on page 345.
Allowable values:	ENABLED, DISABLED, RELEASED
Default value:	RELEASED

Extensible Methods

gxl-control-button::gxl-reflect-selection-state-in-button

gxl-control-button::gxl-set-button-state

gxl-control-button::gxl-perform-button-function

gqsv-single-row-button

This class can be used as a parent class for custom buttons on queue views. This class defines the methods gxl-set-button-state and gxl-reflect-selection-state-in-button so that the button is active when exactly one row is selected in the view. If you subclass from this class, your icon must include the regions icon-highlight and icon-symbol. See "gqsv-multiple-row-button" on page 343 and "gqsv-toolbar-button" on page 345. for alternative parent classes.

Class Inheritance Path

gqsv-single-row-button, gqsv-toolbar-button, gxl-toolbar-button, gxl-control-button, object, item

Attributes

Attribute	Description
gxl-help-text	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	UNSPECIFIED
gxl-help-resource	See "gqsv-toolbar-button" on page 345.
Allowable values:	Inherited
Default value:	GQSV-TEXT-RESOURCES
gxl-button-state	See "gqsv-toolbar-button" on page 345.
Allowable values:	ENABLED, DISABLED, RELEASED
Default value:	RELEASED

Extensible Methods

gxl-control-button::gxl-reflect-selection-state-in-button

gxl-control-button::gxl-set-button-state

gxl-control-button::gxl-perform-button-function

gqsv-toolbar-button

Use this class as the parent for custom buttons you want to include on queue views. You add functionality to your custom button by defining the methods gxl-perform-function, gxl-reflect-selection-state-in-button, and gxl-set-button-state. If you subclass from this class, your icon must include the regions icon-highlight and icon-symbol. For more details, see the *GXL User's Guide*. Alternatively, you can subclass "gqsv-single-row-button" on page 344 or "gqsv-multiple-row-button" on page 343, which define the button activation behavior based on the number of rows selected.

Class Inheritance Path

gqsv-toolbar-button, gxl-toolbar-button, gxl-control-button, object, item

Attributes

Attribute	Description
gxl-help-text	See gxl-control-button (in the *GXL User's Guide*).
Allowable values:	Inherited
Default value:	UNSPECIFIED
gxl-help-resource	See gxl-control-button (in the *GXL User's Guide*).
Allowable values:	Inherited
Default value:	GQSV-TEXT-RESOURCES
gxl-button-state	See gxl-control-button (in the *GXL User's Guide*).
Allowable values:	ENABLED, DISABLED, RELEASED
Default value:	RELEASED

Extensible Methods

gxl-control-button::gxl-reflect-selection-state-in-button

gxl-control-button::gxl-set-button-state

gxl-control-button::gxl-perform-button-function

Button Procedures

The API gqmv-get-filename-from-button is the only API procedure defined explicitly for the GDA queue modules. To extend button functionality, you define the methods gxl-perform-function, gxl-reflect-selection-state-in-button, and gxl-set-button-state on subclasses of gxl-toolbar-button.

The *GXL User's Guide* contains a description of the API available for writing these methods.

gqmv-get-filename-from-button

Synopsis

gqmv-get-filename-from-button
(*Button*: gqmv-file-button, *Client*: g2-window)
–> *Text*

Argument	Description
Button	A button with a filename template, from which a filename is to be generated.
Client	The client for this call.

Return Value	Description
Text	The generated filename.

Description

If a filename has already been created for this button, that filename is returned. Otherwise, a filename is generated using the template and the working directory. The generated filename is presented to the user for approval or editing. That edited filename is then returned.

Extensible Methods

This section describes the queue methods that can be extended by the user.

gda-alarm-entry::gda-update-existing-alarm-entry

Synopsis

gda-alarm-entry::gda-update-existing-alarm-entry
(*Entry*: gda-alarm-entry, *AlarmData*: structure, *Client*: ui-client-item)

Argument	Description
Entry	The existing alarm entry to be updated with a change in alarm.
AlarmData	See "gda-update-existing-alarm-entry" on page 220 for a description of the structure.
Client	The client for this call.

Description

This method is called when an alarm source which has previously entered alarm, and hence has an existing entry, changes state. The transition can be determined from the gda-transition of *AlarmData*.

This method requires call next method. If you are extending this method for your subclass of gda-alarm-entry, be sure to call next method before any code that may override the default attribute values. Extending this method enables an application programmer to update additional attributes that exist on a subclass of gqm-entry. Typically, if you have extended the gda-alarm-entry::gqm-entry-constructor, you extend this method as well.

gda-alarm-entry::gqm-entry-constructor

Synopsis

gda-alarm-entry::gqm-entry-constructor
(*Entry*: gda-alarm-entry, *Queue:* gda-alarm-queue, *EntryData*: structure, *Client*: object)

Argument	Description
Entry	A newly created entry, whose attributes are to be initialized.
Q	The queue on which the entry is being posted.
EntryData	See "gqm-post-entry" on page 183 for the attributes of this structure.
Client	The client for this call.

Description

This method also calls the method for "gda-entry-with-uuid::gqm-entry-constructor" on page 353, so you can refer to that documentation as well. If you are extending this method for your subclass of gda-alarm-entry, be sure to call next method before any code that may override the default attribute values.

The data which *Entry* is populated with either from *EntryData* or from the gdl-alarm-source given by the gqm-source of *EntryData*. One exception is the acknowledged of *Entry*. This attribute is assigned the opposite of the value of the gda-require-acknowledgement of *Entry*. In other words, if alarm entries do not require acknowledgement, that acknowledged attribute is set as true upon construction.

This method should never be called directly.

gda-alarm-entry::gqm-expire-entry

Synopsis

gda-alarm-entry::gqm-expire-entry
(*Entry*: gda-alarm-entry, *Queue*: gda-alarm-queue,
Alarm: gdl-alarm-source, *Client*: object)

Argument	Description
Entry	The alarm entry to expire.
Queue	The alarm queue.
Alarm	The alarm source.
Client	The client or window.

Description

This method essentially reimplements gqm-entry::gqm-expire-entry, but adds a step where the expiration event is recorded to the history of *Entry*. See "gqm-entry::gqm-expire-entry" on page 360 for notes on extending this method.

This method does not use call next method, because the history must be built after the wait, but before the *Entry* is removed from the *Queue*.

gda-alarm-queue::gqs-remove-items

Synopsis

gda-alarm-queue::gqs-remove-items
(*Queue*: gda-alarm-queue, *ItemsToRemove*: item-list, *Client*: object)

Argument	Description
Queue	The queue containing items to be removed.
ItemsToRemove	A list of items to be removed from *Queue*.
Client	The client for this call.

Description

The Alarm queue does not remove unacknowledged entries. If an entry is unacknowledged, that entry remains on the queue. However, the relationship that declares the entry to be "an active entry" of its alarm is broken. See also "gqm-queue::gqs-remove-items" on page 366.

gda-entry-with-uuid::gqm-entry-constructor

Synopsis

gda-entry-with-uuid::gqm-entry-constructor
(*Entry*: gda-entry-with-uuid, *Queue*: gqm-queue, *EntryData*: structure, *Client*: object)

Argument	Description
Entry	The entries of this class are the superior for gda-alarm-entry and gda-explanation-entry. This class should not be access directly, nor extended directly.
Queue	The queue to which the entry was posted.
EntryData	See gqm-post-entry for a description of this structure.
Client	The client for the call.

Description

Although this class is not used by the application programmer, this method is essentially the method for gda-explanation-entry. In addition to calling gqm-entry::gqm-entry-constructor, this method calls gqm-expire-entry if the gqm-queue attribute, gqm-entry-lifetime, is greater than zero. If you have a subclass of gda-explanation-entry for which you are extending this method, make sure you call next method before overriding any attributes.

gda-explanation-entry::gqm-save-entry

Synopsis

gda-explanation-entry::gqm-save-entry
(*Entry*: gda-explanation-entry, *Queue*: gqm-queue, *Filename*: text, *Client*: object)

Argument	Description
Entry	The entry to be written to a file.
Queue	The queue initiating the save.
Filename	The file to which the entry is being saved.
Client	The client for this call.

Description

See "gqm-entry::gqm-save-entry" on page 362 for instructions on extending this method.

gda-recurring-alarm-entry::gqm-entry-constructor

Synopsis

gda-recurring-alarm-entry::gqm-entry-constructor
(*Entry*: gda-recurring-alarm-entry, *Queue*: gda-alarm-queue,
EntryData: structure, *Client*: object)

Argument	Description
Entry	The newly created entry whose attributes are to be populated.
Queue	The queue on which the entry is being posted.
EntryData	See "gqm-post-entry" on page 183 for the details of this structure.
Client	The client for this call.

Description

This is an extension of the gda-alarm-entry::gqm-entry-constructor method, which deals with the additional attribute, gda-recurrances (*recurrences* is misspelled in the code). It also specializes the text of the alarm entry to indicate that this is a recurring alarm. See the parent method for instructions on extending this method.

gdl-alarm-source::gda-acknowledge-alarm

Synopsis

gdl-alarm-source::gda-acknowledge-alarm
(*Alarm*: gdl-alarm-source, *AckTime*: float, *Client*: ui-client-item)

Argument	Description
Alarm	The alarm to be acknowledged.
AckTime	The time of acknowledgement, in G2 time.
Client	The client for this call.

Description

As well as changing the acknowledged attribute on *Alarm*, the base method sets the proper status, acknowledges all connecting entries, logs the alarm summary, and informs all queues of the change in state. You need to call next method to prevent normal alarm behavior from being disabled.

gdl-object::gdl-configure

The gdl-configure method is used to implement the configuration dialogs. It is used to create any additional relations, initialization, or structures that are required when a block is reconfigured. It is not necessary to actually conclude the values into the attributes of your custom block; that is accomplished via call next method. Generally, you would use call next method first, so that the attributes of your block are up to date when the reconfiguration takes place.

Synopsis

gdl-object::gdl-configure
(*CustomBlock*: class gdl-object, *Attributes*: sequence,
Client: class ui-client-item)

Argument	Description
CustomBlock	The Custom Block subclass that you are extending.
Attributes	The sequence of structures describing the attributes to be configured.
Client	The client for this call.

Description

Attributes is a sequence of structures. Each structure represents an attribute that has changed. The structure has attributes attribute-name, the name of the attribute on *CustomBlock*, and attribute-value, the value to be concluded. If gdl-configure is called with an empty *Attributes* sequence, that implies that any or all of the attributes on *CustomBlock* may have been changed previous to the call, so all reconfiguration of the block is to take place.

You can call gdl-configure directly.

gqm-entry::gqm-delete

Synopsis

gqm-entry::gqm-delete
(*Entry*: gqm-entry, *Client*: object)

Argument	Description
Entry	The entry to be deleted.
Client	The client for this call.

Description

This is an extensible method to allow for cleanup of gqm-entry subclasses. Be sure to use call next method to complete the deletion. Specialized code should precede the call next method.

This procedure should never be called by the application developer.

gqm-entry::gqm-entry-constructor

Synopsis

gqm-entry::gqm-entry-constructor
(*Entry*: gqm-entry, *Queue*: gqm-queue, *EntryData*: structure,
Client: object)

Argument	Description
Entry	A newly created error entry whose attributes must be populated with data.
Queue	The queue to which *Entry* is being posted.
EntryData	A structure containing data for *Entry*. See gqm-post-entry for a description of this structure.
Client	The client for this call.

Description

Constructs a new gqm-entry by filling in the gqm-message-text, the gqm-creation-time, and the gqm-priority. The text of the message is obtained from the gqm-text of *EntryData*. If the gqm-text is omitted, the method will generate an error. The gqm-priority is obtained from the gqm-queue attribute, gqm-default-priority. This method is called from gqm-post-entry with a new entry created by the *Queue*. This method should not be called directly.

Extending this method enables an application programmer to initialize additional attributes that exist on a subclass of gqm-entry. Typically, having more attributes to initialize requires that additional data be passed to the gqm-post-entry call, but data could also come from the entry, the queue, or other parts of the system. Be sure to include call next method. The call next method line should precede any code that might override the default value of attributes.

gqm-entry::gqm-expire-entry

Synopsis

gqm-entry::gqm-expire-entry
(*Entry*: gqm-entry, *Queue*: gqm-queue, *Source*: item, *Client*: object)

Argument	Description
Entry	The entry to expire.
Queue	The queue to which the entry was posted.
Source	The source item of the entry, if applicable.
Client	The client or window.

Description

Gqm-expire-entry is called from an entry constructor. The procedure waits for the lifetime specified on the gqm-queue attribute, gqm-entry-lifetime, then removes *Entry* from all queues upon which it resides. Extend this method if you require additional behavior when an entry expires beyond that which takes place with a normal removal.

gqm-entry::gqm-log-entry

Synopsis

gqm-entry::gqm-log-entry
(*Entry*: gqm-entry, *Queue*: gqm-queue, *LogMgr*: gqm-logging-manager, *Client*: object)

Argument	Description
Entry	The queue.
Queue	The queue for which logging is enabled. Logging occurs on a per queue basis.
LogMgr	The logging manager for *Queue*.
Client	The client or g2-window for the call.

Description

This method is called to log a queue entry. It is called by the gqm-queue::gqs-receive-items method if the queue is being logged (see "gqm-logging" on page 381). The method creates the text of the log entry and then logs it with a call to glf-write-to-log-file. To override this method, you can create your own log entry text, then call "glf-write-to-log-file" on page 268. You should not use call next method.

Before subclassing gqm-entry to extend this method, refer to the documentation on "gqm-logging-manager" on page 251 for a discussion on the logging mechanisms. Also, note that this method should generally not be subclassed for gda-alarm-entry. Instead, refer to the gda-alarm-queue-setting attribute gda-alarm-log-formatter.

gqm-entry::gqm-save-entry

Synopsis

gqm-entry::gqm-save-entry
(*Entry*: gqm-entry, *Queue*: gqm-queue, *Filename*: text, *Client*: object)

Argument	Description
Entry	The entry to save.
Queue	The queue requesting that the entry be saved.
Filename	The file to which to save the entry.
Client	The client for this call.

Description

This method is called when the data from an entry is to be saved to a file. Typically this method would be called from a gqmv-save-selected-button of a view of *Queue*, although gqm-save-entry is also an API function.

To extend this method, you must reopen the file named by *FileName*, either before or after the call next method. Alternatively, you could override the format of the logging by writing the data from *Entry* in your own format, without a call next method.

gqm-error-entry::gqm-delete

Synopsis

gqm-error-entry::gqm-delete
(*Entry*: gqm-error-entry, *Client*: object)

Argument	Description
Entry	The error queue to be deleted.
Client	The client for this call.

Description

This method extends gqm-entry::gqm-delete by clearing the block error on any gdl-object which is the-gqm-source-of of *Entry*. Be sure to use call next method after any specialized code if you extend the gqm-delete method for a subclass of gqm-error-entry.

This method should never be called directly.

gqm-error-entry::gqm-entry-constructor

Synopsis

gqm-error-entry::gqm-entry-constructor
(*Entry*: gqm-error-entry, *Queue*: gqm-queue, *EntryData*: structure,
Client: object)

Argument	Description
Entry	A newly created error entry whose attributes must be populated with data.
Queue	The queue to which this entry was posted.
EntryData	A structure consisting either of gqm-text (text) or gqm-error (error).
Client	The window or client object.

Description

This method extends gqm-entry::gqm-entry-constructor. For class gqm-error-entry, the gqm-text of *EntryData* is used if it exists, otherwise this method uses the gqm-error of *EntryData* to create the text of *Entry*.

gqm-queue::gqs-receive-items

Synopsis

gqm-queue::gqs-receive-items
(*Queue*: gqm-queue, *Sender*: item-or-value, *Entries*: item-list,
Client: object)

Argument	Description
Queue	The queue receiving items.
Sender	Typically false. Not used by this subclass.
Entries	A list of the new entries to be added to the queue.
Client	The client for this call.

Description

GQM extends the GQS behavior by adding three functions.

1. First, if the gqm-queue is set up to be logging entries, the entry is logged. This is done by making a call to the method gqm-log-entry on the class of the entry received.
2. Second, the procedure compares the number of items the queue will have at the end of the receiving process with the limit set for the queue in gqm-entry-limit. If the limit is exceeded, the entries are forced into the queue and the oldest entries thrown out.
3. Finally, if the gqm-messages-display for *Queue* is true, the receipt of new entries launches a new view on any *Client* that does not already have a view. This means that if a queue exists, but is hidden behind other workspaces, it is not shown.

gqm-queue::gqs-remove-items

Synopsis

gqm-queue::gqs-remove-items
(*Queue*: gqm-queue, *Items*: item-list, *Client*: object)

Argument	Description
Queue	The queue containing items to be removed.
Items	A list of items to be removed from *Queue*.
Client	The client for this call.

Description

The GQM version of this method actually deletes the entry objects upon removal, assuming the object is not currently residing on another queue. The deletion is carried out in a separate thread. This enables any updating of queues that is to take place in the current thread to complete before the actual deletions occur.

gqmv-tabular-view-template::gqs-create-view

Synopsis

gqmv-tabular-view-template::gqs-create-view
(*Template*: gqmv-tabular-view-template, *Queue*: gqm-queue,
Client: object)
–> *gqmv-Tabular-View-Manager*

Argument	Description
Template	The GQMV tabular view template or subclass that specifies the view format.
Queue	The queue to which the view is to be linked.
Client	The window where the view is to be shown.

Return Value	Description
gqmv-Tabular-View-Manager	The GQMV manager for the view.

Description

This method extends the GQSV version by knowing about the existence of a name panel and the queue count monitors.

Any G2 item can be used as a *Template* provided that the item implements the method gqs-create-view. The method must return a gqs-view-manager or subclass. Subclasses that do not inherit from gqmv-tabular-view-template should not call next method.

In GQM, only one view of a gqm-queue should exist on any given window. Therefore, creating a new view always first checks for the existence of an associated view on *Client* and merely re-shows that view if it already exists.

If gqmv-tabular-view-template is extended, this method can be extended. Be sure to include a call next method statement.

gqs-filter::gqs-apply-filter

Synopsis

gqs-filter::gqs-apply-filter
(*Filter*: gqs-filter, *SourceList*: item-list, *ResultList*: item-list, *Client*: object)

Argument	Description
Filter	The filter that is to be applied to *SourceList*.
SourceList	The input list of items to be filtered.
ResultList	An output list of items that pass the filter criterion.
Client	The client for this call.

Description

Extend this method for subclasses of gqs-filter to create your own filtering algorithm.

gqs-queue::gqs-clear-queue

Synopsis

gqs-queue::gqs-clear-queue
(*Queue*: gqs-queue, *Client*: object)

Argument	Description
Queue	The queue to be cleared.
Client	The client for this call.

Description

Use this procedure to remove all items from a queue. Views, if any, are updated automatically.

If you want to add side effects when all items are removed from a queue, you should subclass gqs-queue and override this method. Be sure to call this method through a call next method statement in your method.

gqs-queue::gqs-receive-items

Synopsis

gqs-queue::gqs-receive-items
(*Queue*: gqs-queue, *Sender*: item-or-value, *IncomingItems*: item-list,
Client: object)

Argument	Description
Queue	The queue receiving items.
Sender	An item indicating the source of the items, or false.
IncomingItems	The items to be added to the queue.
Client	The client for this call.

Description

Call this method to add a list of items to a queue.

If you want certain side effects to occur when items are added to a queue, you can subclass gqs-queue and override this method. If so, be sure to call this method using a call next method statement.

Note that calling this method does not immediately add the items to the queue; instead, it adds them to the queue's input buffer for later insertion.

For more information, see "gqs-force-input-buffer-into-queue" on page 190. This method does use the *Sender* argument, which is provided for user overrides of this method.

gqs-view-manager::gqs-update-view-per-addition

Synopsis

gqs-view-manager::gqs-update-view-per-addition
(*Manager*: gqs-view-manager, *Queue*: gqs-queue, *Items*: item-list, *Client*: object)

Argument	Description
Manager	The manager to be updated
Queue	The underlying queue.
Items	The list of items added to the queue.
Client	The client for this call.

Description

This is a prototype for a method that must be implemented by a subclassed view manager, to update the view in response to one or more items being added to the queue. Since gqs-view-manager is an abstract class, this method should never be called.

gqs-view-manager::gqs-update-view-per-attribute

Synopsis

gqs-view-manager::gqs-update-view-per-attribute
(*Manager*: gqs-view-manager, *Queue*: gqs-queue, *MonitoredItem*: item, *Attribute*: symbol)

Argument	Description
Manager	The manager to be updated
Queue	The underlying queue.
MonitoredItem	The item whose attribute value changed.
Attribute	The name of the attribute.

Description

This is a prototype for a method that must be implemented by a subclassed view manager, to update the view in response to an item contained by the queue receiving a new attribute value. Since gqs-view-manager is an abstract class, this method should never be called.

gqs-view-manager::gqs-update-view-per-delete

Synopsis

gqs-view-manager::gqs-update-view-per-delete
(*Manager*: gqs-view-manager, *Queue*: gqs-queue, *Client*: object)

Argument	Description
Manager	The manager to be updated
Queue	The underlying queue.
Client	The client for this call.

Description

This is a prototype for a method that must be implemented by a subclassed view manager, to update the view in response to one or more items contained by the queue being deleted. Since gqs-view-manager is an abstract class, this method should never be called.

gqs-view-manager::gqs-update-view-per-removal

Synopsis

gqs-view-manager::gqs-update-view-per-removal
(*Manager*: gqs-view-manager, *Queue*: gqs-queue, *Items*: item-list, *Client*: object)

Argument	Description
Manager	The manager to be updated
Queue	The underlying queue.
Items	The list of items removed from the queue.
Client	The client for this call.

Description

This is a prototype for a method that must be implemented by a subclassed view manager, to update the view in response to one or more items being removed from the queue. Since gqs-view-manager is an abstract class, this method should never be called.

Queue Functions

This section describes the functions that support the operation of the queues in GDA.

gda-get-alarm-advice

Synopsis

gda-get-alarm-advice
(*entry*)

Argument	Description
entry	A gda-alarm-entry.

Description

Returns the advice of the entry as a text.

gda-get-alarm-severity

Synopsis

gda-get-alarm-severity
(*entry*)

Argument	Description
entry	A gda-alarm-entry.

Description

Returns the severity of the entry.

gda-get-entry-collection-time

Synopsis

gda-get-entry-collection-time
(*entry*)

Argument	Description
entry	A gda-alarm-entry.

Description

Returns the collection time of the entry as a float.

gqm-get-entry-creation-time

Synopsis

gqm-get-entry-creation-time
(*entry*)

Argument	Description
entry	A gqm-entry.

Description

Returns the creation time of the entry.

gqm-get-entry-message-text

Synopsis

gqm-get-entry-message-text
(*entry*)

Argument	Description
entry	A gqm-entry.

Description

Returns the message text of an entry. Equivalent to the gqm-message-text of *entry*.

gqm-logging

Synopsis

gqm-logging
(*queue*)

Argument	Description
queue	A gqm-queue.

Description

Returns a truth-value indicating whether logging is enabled for that queue.

gqmv-specific-view-on-window

Synopsis

gqmv-specific-view-on-window
(*queue*, *vmclass*, *win*)

Argument	Description
queue	A gqm-queue.
vmclass	A gqs-view-manager class.
win	A g2-window.

Description

Returns a truth-value indicating whether there is a view of a particular class open on the window.

gqmv-view-on-window

Synopsis

gqmv-view-on-window
(*queue*, *win*)

Argument	Description
queue	A gqm-queue.
win	A g2-window.

Description

Returns a truth-value indicating whether there is a view of the queue open on the window.

gqs-number-of-collected-items

Synopsis

gqs-number-of-collected-items
(*queue*)

Argument	Description
queue	An instance of a gqs-queue.

Description

This function returns the number of items contained by a queue.

gqsv-number-of-viewed-items

Synopsis

gqsv-number-of-viewed-items
(*manager*)

Argument	Description
manager	The manager associated with a queue view.

Description

This function enables you to determine the number of items currently contained in a queue view. Because of view filtering and latencies, this is not always the same number of items contained by the underlying queue.

gqsv-view-is-locked

Synopsis

gqsv-view-is-locked
(*item*)

Argument	Description
item	A gxl-spreadsheet, gxl-spreadsheet-view, or gqs-view-manager.

Description

This function enables you to determine if a view is locked, given either the spreadsheet, spreadsheet view, or view manager associated with the view.

Relations Used in Support of the Queues and Views

This section describes the relations used in the queue system.

a-gda-active-entry-of

Properties

Property	Value
inverse	the-gda-active-alarm-of
first class	gda-alarm-entry
second class	gdl-alarm-source
relation type	many-to-one
symmetric?	false
permanent?	false

Description

Denotes that the related entry is active (the alarm is active but the entry has not been acknowledged). See also “the-gqm-source-of” on page 392.

gqmv-view-manager-of-window

Properties

Property	Value
inverse	gqmv-window-of-view-manager
first class	gqs-view-manager
second class	g2-window
relation type	many-to-one
symmetric?	false
permanent?	false

Description

Links a view manager to a g2-window.

gqs-view-manager-for-queue

Properties

Property	Value
inverse	gqs-queue-being-viewed-by
first class	gqs-view-manager
second class	gqs-queue
relation type	many-to-one
symmetric?	false
permanent?	false

Description

This relation enables you to navigate from a queue to its view managers, or vice-versa.

gqsv-spreadsheet-for

Properties

Property	Value
inverse	gqsv-tabular-view-manager-for
first class	gxl-spreadsheet
second class	gqsv-tabular-view-manager
relation type	one-to-one
symmetric?	false
permanent?	false

Description

This relation enables you to find the GXL spreadsheet that is associated with a given view manager, or retrieve the view manager from within a GXL callback procedure on the view.

the-gqm-source-of

Properties

Property	Value
inverse	a-gqmv-entry-of
first class	item
second class	gqm-entry
relation type	one-to-many
symmetric?	false
permanent?	false

Description

Links a gqm-entry with its source item.

the-gqmv-detail-editor-of

Properties

Property	Value
inverse	the-gqmv-detail-source-of
first class	gqmv-composition-box
second class	gqm-view
relation type	one-to-one
symmetric?	false
permanent?	false

Description

This relation enables the master gqm-view to be located, given a gqmv-composition-box which has been launched to edit the comments of an entry.

the-gqmv-detail-view-of

Properties

Property	Value
inverse	the-gqmv-detail-entry-of
first class	gqmv-view
second class	gqm-entry
relation type	many-to-one
symmetric?	false
permanent?	false

Description

When a detail view has been launched for an entry, this relation enables navigation between the two.

Additional Procedures

The procedures in this section are neither callable as an API for the queues, nor are they extensible methods, yet they are referenced elsewhere in this chapter.

These may be API procedures for other parts of GDA, or signatures that are provided as templates.

glf-default-log-file-scheduler

Synopsis

glf-default-log-file-scheduler
(*Log*: glf-logging-manager, *Client*: object)
–> *Integer*

Argument	Description
Log	The logging manager whose closing is to be scheduled.
Client	The client for this call.

Return Value	Description
Integer	For a file opened at the time of the call to this procedure, returns the time to close the file.

Description

This procedure should not be called directly by the user. The default procedure uses the glf-time-interval-to-open-new-log-file and any time entered via the API glf-set-fixed-log-closing-times and returns the nearest file closing time.

If you override the default procedure, you may be disabling these provided ways of configuring the log file scheduling.

gqm-default-ordination

Synopsis

gqm-default-ordination
(*anumber*: integer)
–> *text*

Arguments	Description
anumber	The number.

Return Value	Description
text	The ordinal.

Description

Returns an ordinal as text given an integer. For example, an input of ‘1’ returns ‘1st’, an input of ‘2’ returns ‘2nd’, and so on.

This procedure is also callable as an API to generate ordinal texts as part of your code.

gqsv-highlight-item

Synopsis

gqsv-highlight-item
(*item*: item, *win*: g2-window)

Argument	Description
item	The item to be highlighted.
win	The window where the item is to be shown.

Description

This is the default item highlighter used by the gqsv-go-to-source-button. It shows the item at the center of the screen and creates a flashing arrow that persists for 10 seconds.

If the item is not on a workspace, an error is signaled.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W