G2

System Procedures Reference Manual

Version 2020

G2 System Procedures Reference Manual, Version 2020

June 2020

Ignite Technologies, Inc.
401 Congress Ave., Suite 2650
Austin, TX 78701 USA
Telephone: +1-800-248-0027
Email: success@ignitetech.com

Part Number: DOC013-1200

Contents Summary

Contents

Preface

Describes this document and the conventions that it uses.

About this Manual

This reference manual contains complete information about G2 system procedures. It:

- Introduces G2 system procedures.
- Shows you how to make them available in a knowledge base (KB).
- Describes techniques you can use to view, configure, and invoke them.
- Shows you how to use each system procedure.

Audience

This guide is written for G2 application developers and system integrators who need to use any capability that G2 system procedures provide.

The guide provides simple examples that illustrate systems procedures from within a procedure or method. However, we recommend that you know how to write and use G2 procedures before using the procedures. The necessary information appears in Procedures in the *G2 Reference Manual*.

Organization

The system procedures are grouped into categories. The topics appear alphabetically by category name. The procedures described in each category appear alphabetically by procedure name.

Chapter	Description
Introduction to System Procedures	Introduces G2 system procedures and describes techniques for using them.
Application Deployment Operations	Describes procedures that allow you to automate much of the process of removing source code, of making workspaces proprietary, and of flushing a change log and version information.
Attribute Information Operations	Describes procedures that obtain item and attribute descriptions and support the attribute access facility.
Color Change Operations	Describes procedures for controlling the colors of items on workspaces.
Connection Operations	Describes procedures that copy connections, and get information about them.
Directory Operations	Describes procedures for accessing and changing the current directory.
Error and Message Handler Operations	Describes procedures for handling error messages, operator logbook messages, and message board messages within procedures and methods.
File Operations	Describes procedures that provide access to and management of files external to a KB.
Get Hierarchy Operations	Describes procedures that obtain KB hierarchies.
G2 Graphical Language (G2GL)	Describes the system procedure for using the G2 Graphical Language (G2GL).

Chapter	Description
Hash Table Operations	Describes system procedures for operating on hash tables, which are items that store key-value pairs for fast lookup.
History Clearing Operation	Describes the system procedure that clears history values.
Indexed Attribute Operations	Describes the procedure that retrieves the class instances with a particular value for an indexed attribute.
Inspect Operations	Describes procedures that give you programmatic access to Inspect commands.
KB and Module Operations	Describes procedures for saving, loading, merging, and warmbooting KBs, and for creating, saving, and deleting modules.
List, Array, and Sequence Operations	Describes procedures for getting list, array, and sequence element positions.
Math Operations	Describes dense G2 arrays, sparse G2 arrays, G2 matrices, and the system procedures that create, manipulate, and access them.
Memory Operations	Describes the procedures that help you manage memory.
Move and Transfer Operations	Describes procedures that gather information about one or more objects for the purpose of moving, transferring, cloning, aligning, and distributing the group.
Movement Limit Operations	Describes procedures for limiting the movement of items on workspaces.
Native Menu System (NMS) API	Describes procedures for programmatically creating native menus in Telewindows.
Network Operations	Describes procedures that obtain information about the network and its status.

Chapter	Description
Object Passing Operations	Describes procedures that support item- and object-passing in your KB.
Priority Queue Operations	Describes procedures for operating on priority queues, which are items that store items and associated priorities for fast lookup.
Process Operations	Describes procedures that obtain information about the G2 process, that set a new password, that register a login handler, and that spawn, interrogate, and delete local and remote processes during KB execution.
Profiling Operations	Describes procedures that control profiling during KB execution.
Publish/Subscribe Operations	Describes procedures that allow you to subscribe to G2 item events.
Rule Operations	Describes procedures that get information about forward and backward chaining to rules.
Sorting Operations	Describes procedures that sort array and list elements.
Sound Generation Operation	Describes the procedure that beeps the terminal displaying a specified G2 window.
Time Information Operations	Describes procedures that access the system time and convert time-value formats.
User and Security Information Operations	Describes the system procedures that allow you to access and change G2 user and security data.

Chapter	Description
User Interface Operations	Describes procedures for creating and managing Windows user interface components in the Telewindows client, and for controlling workspace and item appearance, obtaining information about events, and performing other user interface operations.
Version Control	Describes version control procedures for change log entries and text "diff" operations.
Version Information Operations	Describes the procedures for obtaining version information for your G2 process and your sys-mod KB.
Web Operations	Describes the procedures for networking and integration with Web services, SOAP, and HTTP.
XML Parser API	Describes procedures for parsing XML files.
Appendix A, Obtaining the G2-Window of an Item	Describes techniques for mapping an item to its G2-window.
Appendix B, Chart Properties and Enumeration Values	Describes the known chart view properties and enumeration values.
Glossary	

Conventions

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

Typographic

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

Convention Examples	Description
`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 Users' Guide*
- *Business Rules Management System User's Guide*
- *G2 Reporting Engine User's Guide*
- *G2 Web User's Guide*
- *G2 Event and Data Processing User's Guide*
- *G2 Run-Time Library User's Guide*
- *G2 Event Manager User's Guide*
- *G2 Dialog Utility User's Guide*
- *G2 Data Source Manager User's Guide*
- *G2 Data Point Manager User's Guide*
- *G2 Engineering Unit Conversion User's Guide*
- *G2 Error Handling Foundation User's Guide*
- *G2 Relation Browser User's Guide*

Bridges and External Systems

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

G2 JavaLink

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

G2 Diagnostic Assistant

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

Customer Support Services

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

To obtain customer support online:

➔ Access Ignite Support Portal at `https://support.ignitetech.com`.

You will be asked to log in to an existing account or create a new account if necessary. Ignite Support Portal 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 or email:

➔ Use the following numbers and addresses:

United States Toll-Free	+1-855-453-8174
United States Toll	+1-512-861-2859
Email	support@ignitetech.com

1

Introduction to System Procedures

Introduces G2 system procedures and describes techniques for using them.

Introduction

G2 provides many system-defined procedures that let you perform a variety of operations. These procedures are called **system procedures**. The G2 system procedures:

- Provide programmatic access to many G2 capabilities that are also available interactively.
- Provide programmatic access to additional capabilities that are not available interactively.

This chapter:

- Introduces the categories of system procedures.
- Shows you how to make system procedures available in a G2 knowledge base (KB).
- Describes techniques you can use to view and invoke system procedures.
- Lists the system procedures that cause G2 to enter a wait state.

Categories of G2 System Procedures

System procedures fall into categories according to their purposes. Some categories contain only one procedure. The chapters in this manual reflect the system procedure categories. They are listed in the Preface under Organization.

System procedures fall into categories according to their purposes. Some categories contain only one procedure. For a list of all the categories, see Organization.

Making System Procedures Available in a KB

All G2 system procedures exist in a modularized knowledge base (KB), named `sys-mod.kb`, which has a **required module**, `uilroot.kb`. Before you can invoke any system procedure, you must merge `sys-mod.kb` into your KB as a **module**.

Your KB will be inconsistently modularized after merging `sys-mod.kb`. To modularize your KB, make `sys-mod` a required module so that the `sys-mod` and `uilroot` modules are automatically included when you load your KB. For information on modules and how to make your KB consistently modularized, see Modularized KBs in the *G2 Reference Manual*.

Gensym supplies `sys-mod.kb` and `uilroot.kb` in the `kbs/utils` subdirectory of the directory that contains G2. For information on `uilroot.kb`, see the *G2 GUIDE User's Guide*.

To merge the G2 system procedures interactively:

1. Choose Main Menu > Merge KB.
2. Specify `sys-mod.kb` located in the `utils` subdirectory of the `g2` directory.
3. Click End.

Once `sys-mod.kb` has been merged, you can invoke any G2 system procedure.

Viewing System Procedures

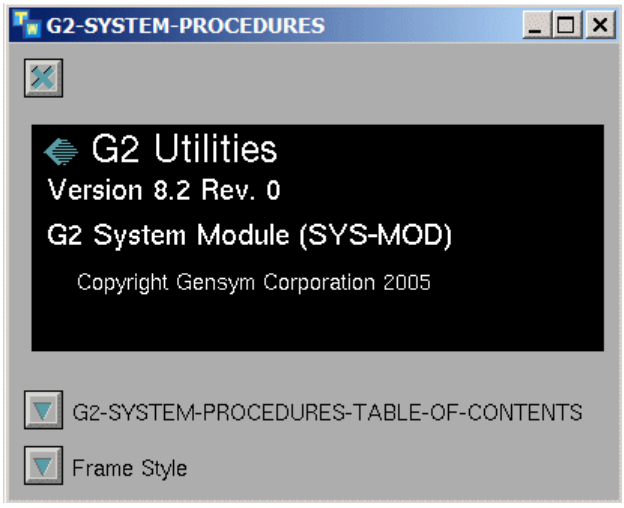
The workspace that contains the system procedures is initially hidden, so merging `sys-mod.kb` does not change the appearance of your KB.

To see the workspace that contains the system procedures:

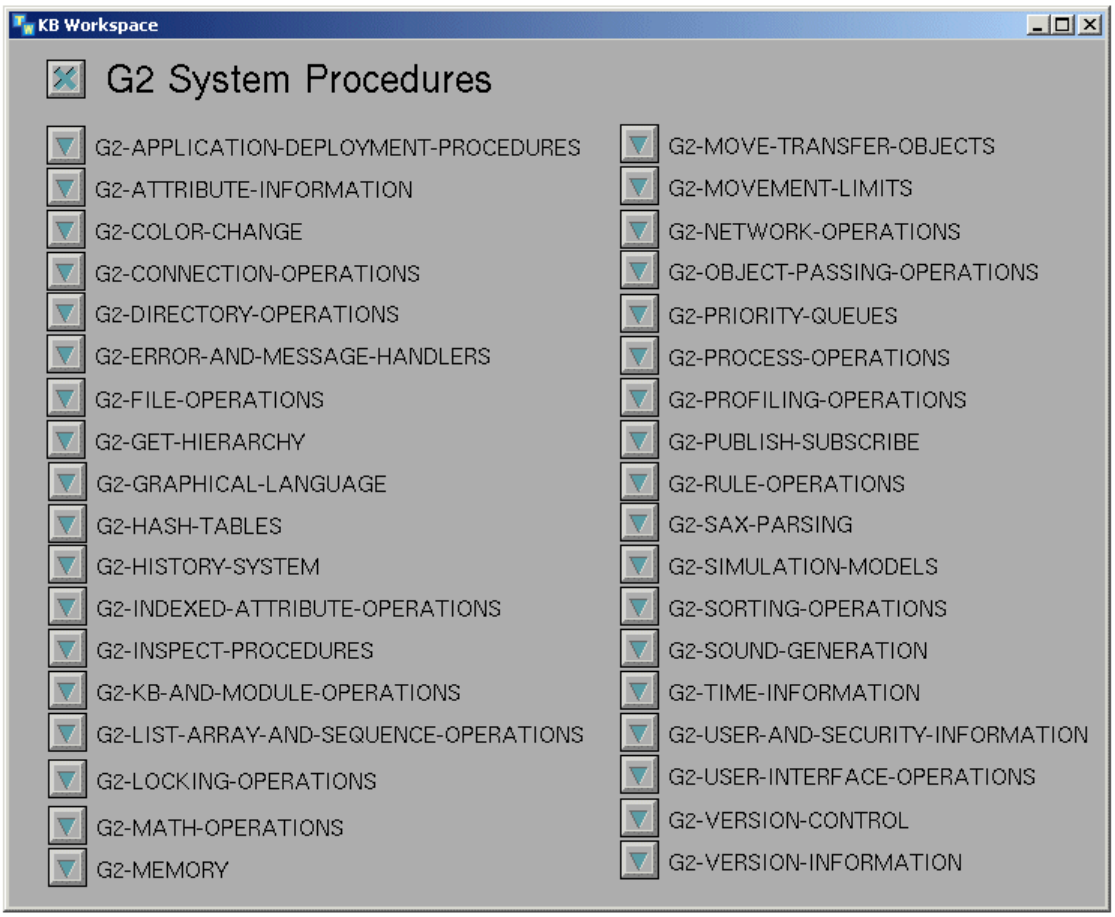
1 Choose Main Menu > Get Workspace.

2 Select g2-system-procedures.

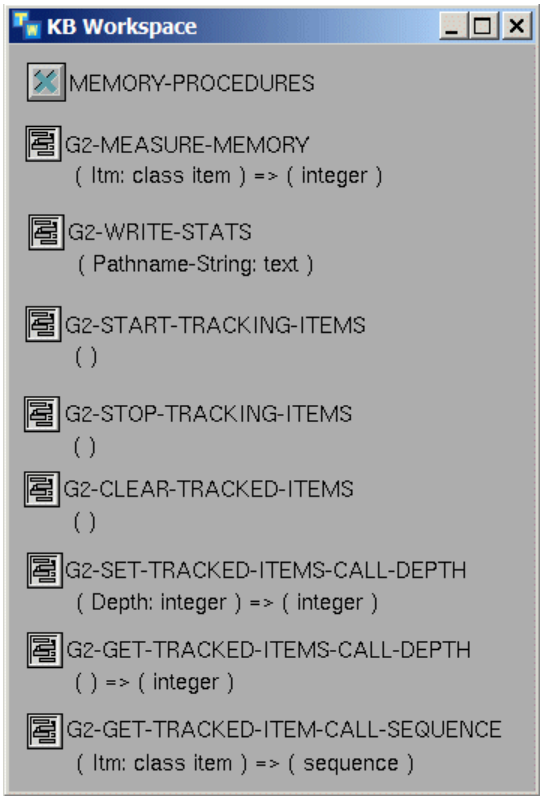
The top-level g2-system-procedures workspace, which includes module and version information, is displayed:



Clicking the g2-system-procedures-table-of-contents button displays a workspace of navigation buttons that take you to the **subworkspaces** of different categories of system procedures:



For example, the subworkspace of the g2-memory navigation button takes you to this subworkspace:



System procedures are similar to ordinary procedures, but they have been text-stripped to prevent modifications that could impair the integrity of G2; therefore, you cannot configure system procedures as you could ordinary procedures. In addition, you cannot delete the system procedure workspaces because they are proprietary.

Invoking System Procedures

Use the call **statement** or the start **action** to invoke a system procedure. For example:

```
call g2-enable-profiling ( )
```

or:

```
start g2-enable-profiling ( )
```

Invoking a system procedure is the same as invoking a user-defined procedure. For detailed information about procedure invocation, see Procedures in the *G2 Reference Manual*.

System Procedures and Wait States

The following system procedures cause G2 to enter a wait state, which allows other processing that can asynchronously change the G2 context. Be sure to revalidate the context as needed after any of these procedures returns. This warning is repeated in the documentation for each of the procedures.

Process Operations

- g2-spawn-remote-process-to-run-command-line
- g2-spawn-remote-process (superseded by the preceding)
- g2-spawn-remote-process-with-arguments
- g2-remote-process-exists
- g2-kill-remote-process

KB and Module Operations

- g2-save-kb
- g2-reclaim-save-module-memory
- g2-save-module

The following system procedures save your KB without interruption:

- g2-save-kb-without-other-processing
- g2-save-module-without-other-processing
- g2-snapshot-without-other-processing

Sorting Operations

- g2-sort-array
- g2-sort-list
- g2-sort (superseded by both of the preceding)

2

Application Deployment Operations

Describes procedures that allow you to automate much of the process of removing source code, of making workspaces proprietary, and of flushing a change log and version information.

Introduction

These procedures allow you to automate much of the process of packaging large KB-based applications for customer deployment, including removing source code, making workspaces proprietary, and flushing a change log and version information.

Note If you wish, you can still *manually* mark items for text-stripping through menu choices and mark workspaces that you want to make proprietary through menu choices. For information on preparing KBs for customer distribution, see Package Preparation in the *G2 Reference Manual*.

Use the application deployment system procedures for:

- Flushing change logs and version information
- Text-stripping KBs programmatically
- Automatically making workspaces proprietary

Flushing Change Logs and Version Information

These system procedures allow you to flush change logs and version information:

g2-flush-change-log-for-entire-kb

g2-flush-change-log-for-item

g2-flush-version-information

g2-flush-change-log-for-entire-kb

Flushes the change log for the entire KB.

Synopsis

g2-flush-change-log-for-entire-kb
( )

Description

Invoking this procedure flushes the change log entries for every module in the current KB, not just the top-level module or any other single module. It is the programmatic equivalent to the Flush Change Log For Entire KB menu choice on the Main Menu > Miscellany menu.

Because flushing a KB of its change log and version information is a permanent and irreversible change, it is accessible only when the KB is in package preparation mode.

Note For more information on flushing change logging and version information from KBs, refer to the "Package Preparation" chapter in the *G2 Reference Manual, Version*.

---

g2-flush-change-log-for-item

Flushes the change log for an item.

Synopsis

g2-flush-change-log-for-item
(*item*:class item)

Argument	Description
item	Any KB item whose user- or system-provided class name you wish to flush from the change log.

Description

Invoking this procedure permanently deletes all entries in the item's change log. It is the programmatic equivalent to the flush-change-log menu choice on the change-log attribute submenu of rules and other definition items.

Because flushing an item of its change log and version information is a permanent and irreversible change, it is accessible only when the KB is in package preparation mode.

Note For more information on flushing change logging and version information from KBs, refer to the "Package Preparation" chapter in the *G2 Reference Manual.*

g2-flush-version-information

Flushes the KB version information from the KB's change log.

Synopsis

g2-flush-version information
(*saving-params*: class saving-parameters)

Argument	Description
saving-params	A saving parameter who version information you wish to remove.

Description

Invoking this procedure permanently deletes all the KB version information from the KB's change log. It is the programmatic equivalent of choosing the KB-version-information-for-change-logging attribute submenu of the Saving Parameters System Table.

Because flushing a KB of its change log and version information is a permanent and irreversible change, it is accessible only when the KB is in package preparation mode.

Note For more information on flushing change logging and version information from KBs, Package Preparation in the *G2 Reference Manual*.

Text-Stripping KBs Programmatically

The following system procedures allow you to programmatically text-strip KBs by providing access to the hidden attribute strip-text-mark/do-not-strip-text-mark.

g2-strip-texts-now

g2-set-strip-text-mark

g2-item-has-strip-text-mark

g2-set-do-not-strip-text-mark

g2-item-has-do-not-strip-text-mark

g2-strip-texts-now

Programmatically strips the text of any items marked for text-stripping.

Synopsis

g2-strip-texts-now
( )

Description

Text stripping means to remove source code from compiled attributes. Upon your invocation of this procedure, G2 strips the text of any items marked for text stripping.

Invoking this procedure is the programmatic equivalent of selecting the Main Menu > Miscellany > Strip Texts Now menu item and clicking OK in the confirmation dialog.

Caution Invoking this system procedure will *not* call up the confirmation dialog. The warning in the dialog still obtains for programmatic operations. Text-stripping KB items or workspaces will remove the text associated with all items marked to be text-stripped. These changes are irreversible in your target KB or workspace.

If you did not save the source KB before marking items to be text-stripped, save it before invoking this procedure. Save the target KB with a different name than the source KB, or save it in a different directory.

The Operator Logbook displays the messages: Stripping texts now! and Finished stripping texts to let you know when G2 has finished. To check the results, check your target KB for confirmation.

Related Procedures

Use this procedure in conjunction with:

- g2-enter-package-preparation-mode
- g2-set-strip-text-mark
- g2-item-has-strip-text-mark
- g2-set-do-not-strip-text-mark
- g2-item-has-do-not-strip-text-mark

You are *not* required to enter Package Preparation Mode when invoking text-stripping system procedures, as you *are required to do* for the corresponding menu commands.

g2-set-strip-text-mark

Programmatically marks KB items or workspaces for text-stripping.

Synopsis

g2-set-strip-text-mark
(*item*: class item, *new-state*: truth-value)

Argument	Description
item	Any KB item whose user- or system-provided class name you wish to mark.
new-state	A truth-value that is true if the item is to be marked for text-stripping, or false if it is not to be marked.

Description

A strip-text mark will also mark for text-stripping every subordinate KB item and subworkspace in the workspace hierarchy of the marked item, and will remove any previous do-not-strip text marks from these items.

Invoking this procedure is the programmatic equivalent of selecting the Main Menu > Miscellany > Mark To Strip Text menu item and clicking OK in the confirmation dialog.

Caution Invoking this system procedure will *not* call up the confirmation dialog. The warning in the dialog still obtains for programmatic operations. Text-stripping KB items or workspaces will remove the text associated with all items marked to be text-stripped. Once you text-strip the marked items, these changes are irreversible in your target KB or workspace.

Save the source KB before text stripping. Save the target KB with a different name than the source KB, or save it in a different directory.

Related Procedures

Use this procedure in conjunction with:

- g2-strip-texts-now
- g2-item-has-strip-text-mark
- g2-set-do-not-strip-text-mark
- g2-item-has-do-not-strip-text-mark

You are *not* required to enter Package Preparation Mode when invoking text-stripping system procedures, as you *are required to do* for the corresponding menu commands.

g2-item-has-strip-text-mark

Checks for items that are marked for text-stripping.

Synopsis

g2-item-has-strip-text-mark
(*item*: class item)
-> *return-value*: truth-value

Argument	Description
item	Any KB item whose user- or system-provided class name you wish to check for a Mark To Strip Text mark.

Return Value	Description
return-value	A truth-value that is true if the item is marked for text-stripping, or false if it is not.

Description

This procedure provides a local check on a KB item itself and only on the item itself.

If a superior object in a workspace hierarchy has been marked for text-stripping, the Mark To Strip Text menu choice of this local KB item will appear with an asterisk next to it.

Related Procedures

Use this procedure in conjunction with:

- g2-strip-texts-now
- g2-set-strip-text-mark
- g2-set-do-not-strip-text-mark
- g2-item-has-do-not-strip-text-mark

You are *not* required to enter Package Preparation Mode when invoking text-stripping system procedures, as you *are required to do* for the corresponding menu commands.

g2-set-do-not-strip-text-mark

Programmatically sets a do-not-strip-text mark on all the items in a workspace.

Synopsis

g2-set-do-not-strip-text-mark
(*item*: class item, *new-state*: truth-value)

Argument	Description
item	Any KB item whose user- or system-provided class name you wish to mark not to be text-stripped.
new-state	A truth-value that is true if the item is to be marked not to be text-stripped, or false if it is not to be so marked.

Description

Invoking this procedure is the programmatic equivalent of selecting the Main Menu > Miscellany > Mark Not To Strip Text menu item and clicking OK in the confirmation dialog.

Caution Invoking this system procedure will *not* call up the confirmation dialog. The warning in the dialog still obtains for programmatic operations. Not text-stripping workspaces will retain the text associated with all items.

Save the source KB before text stripping. Save the target KB with a different name than the source KB, or save it in a different directory.

Related Procedures

Use this procedure in conjunction with:

- g2-strip-texts-now
- g2-set-strip-text-mark
- g2-item-has-strip-text-mark
- g2-item-has-do-not-strip-text-mark

You are *not* required to enter Package Preparation Mode when invoking text-stripping system procedures, as you *are required to do* for the corresponding menu commands.

g2-item-has-do-not-strip-text-mark

Checks for items that are marked not to be text-stripped.

Synopsis

g2-item-has-do-not-strip-text-mark
(*item*: class item)
-> *return-value*: true or false

Argument	Description
item	Any KB item whose user- or system-provided class name you wish to check for a Mark Not To Strip Text mark.

Return Value	Description
return-value	A truth-value that is true if the item is marked not to be text-stripped, or false if it is not so marked.

Description

This procedure provides a local check on an item itself and only the item itself.

If a superior object in a workspace hierarchy has been marked not to be text-stripped, the Mark Not To Strip-Text menu choice of this local KB item will appear with an asterisk next to it.

Related Procedures

Use this procedure in conjunction with:

- g2-strip-texts-now
- g2-set-strip-text-mark
- g2-item-has-strip-text-mark
- g2-set-do-not-strip-text-mark

You are *not* required to enter Package Preparation Mode when invoking text-stripping system procedures, as you *are required to do* for the corresponding menu commands.

Automating Making Workspaces Proprietary

These procedures partially automate the process of making workspaces proprietary:

- g2-make-workspaces-proprietary-now
- g2-enter-package-preparation-mode
- g2-leave-package-preparation-mode
- g2-is-in-package-preparation-mode
- g2-enter-simulate-proprietary-mode
- g2-leave-simulate-proprietary-mode
- g2-is-in-simulate-proprietary-mode

g2-make-workspaces-proprietary-now

Programmatically makes workspaces in your KB proprietary that have been configured with proprietary package names.

Synopsis

g2-make-workspaces-proprietary-now
( )

Description

Invoking this procedure is the programmatic equivalent of selecting the Main Menu > Miscellany > Make Workspaces Proprietary Now menu item and clicking OK in the confirmation dialog.

Caution Invoking this system procedure will *not* call up the confirmation dialog. The warning in the dialog still obtains in programmatic mode. This operation will cause those workspaces that have been given a potential proprietary package attribute to become permanently proprietary to those packages and thus severely restricted in their use. Once this has been done, any KB that has merged in those workspaces will only be allowed to execute with special authorization codes obtained through Gensym Corporation. These changes are irreversible.

Distributing a Proprietary Application Package

If you distribute a proprietary KB to your end users, you must register the name of the application package with Gensym. If the proprietary-package attribute or any workspace specifies the package name (instead of private or none), your end users must modify the `g2.ok` files to be authorized to use the KB.

Related Procedures

Use this procedure in conjunction with:

- g2-enter-package-preparation-mode
- g2-leave-package-preparation-mode
- g2-is-in-package-preparation-mode
- g2-enter-simulate-proprietary-mode
- g2-leave-simulate-proprietary-mode
- g2-is-in-simulate-proprietary-mode

g2-enter-package-preparation-mode

Displays the menu choices for each item in package preparation mode.

Synopsis

g2-enter-package-preparation-mode
( )

Description

Invoking this procedure is the programmatic equivalent of selecting the Main Menu > Miscellany > Enter Package Preparation Mode menu item and clicking OK in the confirmation dialog.

Caution Invoking this system procedure will *not* call up the confirmation dialog. The warning in the dialog still obtains in programmatic mode. This operation can add options to menus that allow for potentially dangerous and irreversible changes.

The Operator Logbook displays the message: "Entering Package Preparation mode" to let you know when you are in package preparation mode.

Note You create proprietary KBs in manual mode. Follow the instruction on how to make proprietary workspaces in the "Package Preparation" chapter in the *G2 Reference Manual.*

Related Procedures

Use this procedure in conjunction with:

- g2-leave-package-preparation-mode
- g2-is-in-package-preparation-mode
- g2-enter-simulate-proprietary-mode
- g2-leave-simulate-proprietary-mode
- g2-is-in-simulate-proprietary-mode
- g2-make-workspaces-proprietary-now

g2-leave-package-preparation-mode

Exits your KB item or workspace from package preparation mode.

Synopsis

g2-leave-package-preparation-mode
( )

Description

Invoking this procedure is the programmatic equivalent of selecting the Main Menu > Miscellany > Leaving Package Preparation Mode menu item and clicking OK in the confirmation dialog.

The Operator Logbook displays the message: Leaving Package Preparation mode to let you know when you have exited package preparation mode.

Related Procedures

Use this procedure in conjunction with:

- g2-enter-package-preparation-mode
- g2-is-in-package-preparation-mode
- g2-enter-simulate-proprietary-mode
- g2-leave-simulate-proprietary-mode
- g2-is-in-simulate-proprietary-mode
- g2-make-workspaces-proprietary-now

g2-is-in-package-preparation-mode

Checks to see that your G2 is in package preparation mode.

Synopsis

g2-is-in-package-preparation-mode
( )
-> *return-value*: truth-value

Return Value	Description
return-value	A truth-value that is true if the G2 is in package preparation mode, or false if it is not.

Related Procedures

Use this procedure in conjunction with:

- g2-enter-package-preparation-mode
- g2-leave-package-preparation-mode
- g2-enter-simulate-proprietary-mode
- g2-leave-simulate-proprietary-mode
- g2-is-in-simulate-proprietary-mode
- g2-make-workspaces-proprietary-now

g2-enter-simulate-proprietary-mode

Allows you to simulate the behavior of your target proprietary KB or workspace.

Synopsis

g2-enter-simulate-proprietary-mode
( )

Description

Invoking this procedure is the programmatic equivalent of choosing the Main Menu > Miscellany > Enter Simulate Proprietary Mode menu item.

This option simulates the behavior of your KB as if your computer was authorized to use the named package. You can enter and leave simulate proprietary mode as many times as you need to change the KB behavior until you are satisfied.

To exit from the simulated proprietary mode, invoke g2-leave-simulate-proprietary-mode.

Tip You do not have to be running the KB in package preparation mode to simulate proprietary mode. You can simulate proprietary mode at any time.

Related Procedures

Use this procedure in conjunction with:

- g2-enter-package-preparation-mode
- g2-leave-package-preparation-mode
- g2-is-in-package-preparation-mode
- g2-leave-simulate-proprietary-mode
- g2-is-in-simulate-proprietary-mode
- g2-make-workspaces-proprietary-now

g2-leave-simulate-proprietary-mode

Exits your target KB or workspace from simulated proprietary mode.

Synopsis

g2-leave-simulate-proprietary-mode
( )

Description

Invoking this procedure is the programmatic equivalent of choosing the Main Menu > Miscellany > Leave Simulate Proprietary Mode menu item.

You can enter and leave simulate proprietary mode as many times as you need to change the KB behavior until you are satisfied.

Tip You do not have to be running the KB in package preparation mode to simulate proprietary mode. You can simulate proprietary mode at any time.

Related Procedures

Use this procedure in conjunction with:

- g2-enter-package-preparation-mode
- g2-leave-package-preparation-mode
- g2-is-in-package-preparation-mode
- g2-enter-simulate-proprietary-mode
- g2-is-in-simulate-proprietary-mode
- g2-make-workspaces-proprietary-now

g2-is-in-simulate-proprietary-mode

Checks that your G2 is in simulate proprietary mode.

Synopsis

g2-is-in-simulate-proprietary-mode
( )
-> *return-value*: truth-value

Return Value	Description
return-value	A truth-value that is true if the G2 is in simulate proprietary mode, or false if it is not.

Description

Tip You do not have to be running the KB in package preparation mode to simulate proprietary mode. You can simulate proprietary mode at any time.

Related Procedures

Use this procedure in conjunction with:

- g2-enter-package-preparation-mode
- g2-leave-package-preparation-mode
- g2-is-in-package-preparation-mode
- g2-enter-simulate-proprietary-mode
- g2-leave-simulate-proprietary-mode
- g2-make-workspaces-proprietary-now

3

Attribute Information Operations

Describes procedures that obtain item and attribute descriptions and support the attribute access facility.

Introduction

Use the attribute information system procedures to obtain detailed information about items and their attributes.

g2-get-all-reserved-system-attribute-names

Returns a sequence of all reserved system-defined class attribute names, in alphabetical order.

Synopsis

g2-get-all-reserved-system-attribute-names
(*type*: symbol)
-> *reserved-words*: sequence

Argument	Description
type	One of the following symbols: • ordinary — Returns all non-hidden attributes. • hidden — Returns all hidden attributes of user-extensible classes. • all — Returns both ordinary and hidden attributes.

Return Value	Description
reserved-words	A sequence of reserved words of the specified *type*.

Description

If *g2-attribute* is the name of a reserved attribute of a system-defined class, then you cannot use it as the name of an attribute of a user-defined class.

Some system-defined classes such as object or connection are user-extensible; other system-defined classes such as logbook-parameters are not user-extensible. To avoid possible inheritance problems, you cannot use system-defined attributes of user-extensible system-defined classes as user-defined attributes; thus, these attributes are considered reserved words in G2. However, you can use system-defined attributes of non user-extensible system-defined classes as user-defined attributes; these attributes are considered unreserved.

If you attempt to use a reserved word as a user-defined attribute, G2 takes the following actions:

- When entering a reserved word in the G2 Text Editor, an error such as the following appears in the text editor:

This is uncompilable. HEIGHT-OF-IMAGE is the name of a G2 system attribute and cannot be a user-defined attribute."

- When loading a KB from an older version of G2 in which the reserved word was not a system-defined attribute, an error such as the following appears in the Operator Logbook:

  HEIGHT-OF-IMAGE is the name of a G2 system attribute and cannot be a user-defined attribute.

  Also, the notes of the user-defined class-definition contains an error such as the following:

  OK, and note that the class-specific-attribute height-of-image is now a reserved G2 attribute. You must rename it before starting G2.

To obtain the name of the user-defined class that uses the reserved word, use the following Inspect command:

highlight the symbol height-of-image in every class-definition

If *g2-hidden-attribute* is the name of a hidden attribute of a system-defined class, you may use it as the name of an attribute of a user-defined class. However, we recommend that you avoid this practice. For example, using history as the name of an attribute of a user-defined class would shadow its use as a hidden attribute of a float-parameter. Similarly, using containing-module as the name of an attribute of a user-defined class would shadow its use in GFR and GMS.

Note that you cannot use reserved symbols as the name of a user-defined attribute.

For a list of reserved symbols, see Appendix B, Reserved Symbols in the *G2 Reference Manual*.

g2-get-attribute-descriptions-of-class

Returns the class attribute descriptions for the attribute names you specify.

Synopsis

g2-get-attribute-descriptions-of-class
(*class-name*: symbol, *specific-attributes*: value)
-> *attribute-descriptions*: sequence

Argument	Description
class-name	A symbol naming the class whose attribute descriptions you wish to obtain.
specific-attributes	A value that can either be a sequence of one or more attributes for which you require attribute descriptions, or the value true. Specify true for this argument to return descriptions for every visible attribute in a class. **Note:** If you are calling this procedure from G2 Gateway, specifying true gives an error. Instead, use an empty sequence.

Return Value	Description
attribute-descriptions	A sequence of structures of one or more attribute descriptions of the attributes provided in the second argument, and described in the next section.

Each structure in *attribute-descriptions* contains the following information about an attribute specified via *specific-attributes*:

Subattribute	Type	Description
attribute-name	symbol	The attribute name.
virtual	truth-value	When false, the attribute has an attribute description; when true, it is a class characteristic exported as a virtual attribute. Both attributes can be accessed with the same syntax.
text-readable	truth-value	Indicates whether you can use the expression: the text of the *attribute-name* of item
text-writable	truth-value	Indicates whether you can set the attribute value, using the action: change the text of the *attribute-name* of item
value-readable	truth-value	Indicates whether you can use the expression: the *attribute-name* of item
value-writable	truth-value	Indicates whether you can set the value of the attribute, using the action: conclude that the *attribute-name* of item...
is-system-defined	truth-value	Indicates whether an attribute is system- or user-defined. This attribute is true for all system-defined classes.
defining-class	symbol	The class in which the attribute is defined, which could be the current class or a superior class.

Depending upon the class and attribute in question, the structure may optionally contain zero or more of the following subattributes:

This subattribute...	Describes...
type	A structure of the valid types for this attribute. A detailed description of this subattribute follows.
initial-value	This field appears if the attribute has a default value that is not a class instance. The field's value is an item-or-value indicating the default value of the attribute.
initial-item-class	This field appears if the attribute has a default value that is a class instance. The field's value is a symbol that names the class.
editor-grammar	This field appears when the attribute is text-writable. It is a symbol indicating the G2 grammar used to edit the text of this attribute.
superseded	Truth-value indicating if the attribute is not recommended for use. If this attribute is true, then the attribute has been superseded by another with similar functionality.

Type Subattribute Definition

The type attribute of the attribute description specifies in detail what values can be given to an attribute. In some cases, the type value is a simple one, such as integer. In other cases, the type value of an attribute can be a complex or composite type consisting of several structures and sequences which, in turn, are combined with one or more Boolean operators.

The only attribute required in an attribute's type structure is type-or-operator. Attributes with a simple value type merely specify the type, integer, float, text, and so on. Complex and composite types require more description and thus more structure subattributes. The specification of the type structure can begin:

type-structure :=
    structure
        (type-or-operator: *type-or-operator-values*,
            [*attribute-name: attribute-value* [,...]])

The possible *type-or-operator-values* are as follows:

value |
symbol |
text |
truth-value |
quantity | integer | float |
or | and |
not |
member |
quantity-range | integer-range | float-range |
structure |
sequence |
class |
none

The rest of the structure may include zero or more of these attributes, depending on the value of type-or-operator:

type-or-operator	**attribute-name**	**attribute-value**
or | and	operands	sequence of type-structure
not	operand	type-structure
member	possible-values	sequence of symbols
quantity-range | integer-range | float-range	high	quantity | integer | float
	low	quantity | integer | float
structure	*slots*	sequence of *structure-slots-desc*
	required-slots	*required-slots-desc*
sequence	element-type	type-structure
	minimum-elements	integer
	maximum-elements	integer
class	class-name	symbol

For the structure-related attributes *slots* and *required-slots*, two more kinds of structures exist:

structure-slots-desc ::=
    structure
        (attribute-name: symbol, type: *type-structure*)

required-slots-desc ::=
    sequence
        (and | or | not, {attribute-name | *required-slots-desc*} [,...])

For more information about using attribute descriptions, see Attribute Access Facility the *G2 Reference Manual*.

Description

This system procedure returns a sequence of structures containing the attribute-descriptions of one or more attributes of a class. If you provide a sequence of one or more symbols of attribute names, the attribute descriptions for those attributes alone are returned. The system-procedure g2-get-attributes-visible-in-mode, returns the attributes that are accessible for a given user mode. Providing the value true, rather than a sequence of attribute names, returns a sequence of attribute descriptions for the entire class.

Example

This example uses a user-defined class named equipment which inherits directly from object. The equipment class defines an attribute named producer; its type is an instance of the class company.

This example procedure call:

```
call g2-get-attribute-descriptions-of-class(the symbol equipment,
    sequence(the symbol containing-module, the symbol producer)
```

returns this sequence of structures:

```
sequence(
    structure(
        attribute-name: the symbol containing-module,
        virtual: true,
        text-readable: false,
        text-writable: false,
        value-readable: true,
        value-writable: false,
        initial-value: none,
        is-system-defined: true,
        defining-class: the symbol item,
        type: structure (type-or-operator: the symbol symbol)),
    structure(
        attribute-name: the symbol producer,
        virtual: false,
        text-readable: true,
        text-writable: true,
        value-readable: true,
        value-writable: true,
        is-system-defined: false,
        defining-class: the symbol equipment,
        initial-item-class: the symbol company,
        editor-grammar: the symbol item-or-datum-attribute,
        type: structure(type-or-operator: the symbol class,
                        class-name: the symbol company)))
```

g2-get-attribute-texts-of-item

Returns the text of one or more attributes.

Synopsis

g2-get-attribute-texts-of-item
(*item*: class item, *specific-attributes*: value)
-> *attribute-texts*: structure

Argument	Description
item	The item whose attribute texts you wish to obtain.
specific-attributes	A value which can either be a sequence of symbols that name one or more attributes whose texts you wish to obtain, or the value true. Specify true for this argument to return the texts of every attribute in a class. **Note:** If you are calling this procedure from G2 Gateway, specifying true gives an error. Instead, use an empty sequence.

Return Value	Description
attribute-texts	A structure consisting of the names of the attributes and their textual representations.

Description

Returns a structure of the text of an item's attributes. As with g2-get-attribute-values-of-item, the returned structure resembles the item itself. Rather than containing values, however, the structure contains text, as though one had iterated over the given attributes of the item with the expression:

conclude that the *attribute-name* of *structure* =
the text of the *attribute-name* of *item*

An attribute name/text pair for each attribute will appear in the returned structure unless one of these conditions is true:

- The attribute is not visible due to proprietary restrictions.
- Only the attribute value, not its text, is accessible. For example, the name-box attribute has a value, but it is neither text-readable, nor text-writable, and thus is not present in the attribute tables.

Example

This example employs an instance of the user-defined class equipment used in the example g2-get-attribute-descriptions-of-class.

This example procedure call:

```
call g2-get-attribute-texts-of-item(equipment101, true)
```

returns this structure:

```
structure(
    uuid: "@"9d00016781a411d38888889a00080c90@"",
    notes: "OK",
    item-configuration: "none",
    names: "EQUIPMENT101",
    producer: "the company COMPANY1",
    size: "medium")
```

g2-get-attribute-values-of-item

Returns the values of one or more attributes of an item.

Synopsis

g2-get-attribute-values-of-item
(*item*: class item, *specific-attributes*: value)
-> *attribute-values*: structure

Argument	Description
item	The item whose attribute values you wish to obtain.
specific-attributes	A value that can either be a sequence of symbols that name one or more attributes whose values you wish to obtain, or the value true. Specify true for this argument to return the value of every visible attribute in an item. **Note:** If you are calling this procedure from G2 Gateway, specifying true gives an error. Instead, use an empty sequence.

Return Value	Description
attribute-values	A structure consisting of the names of the attributes and their values.

Description

Returns a structure of values for the attributes of this item. The structure resembles the item itself: its attribute names and values match the item's attribute names and values, as if one had iterated over the given attributes of the item with an expression such as:

conclude that the *attribute-name* of *structure* = the *attribute-name* of *item*

These values can be of any G2 value types, including sequences and structures.

An attribute/value pair for each attribute will appear in the returned structure unless one of these conditions is true:

- The value of the attribute is none.
- The attribute is not visible due to proprietary restrictions.

Example

This example uses an instance of the user-defined equipment class used in the example for g2-get-attribute-descriptions-of-class.

This example procedure call:

```
call g2-get-attribute-values-of-item(equipment101,
    sequence(the symbol size-in-workspace, the symbol permanent,
                the symbol producer)
```

returns this structure:

```
structure(
    size-in-workspace: structure(width: 50, height: 50),
    permanent: true,
    producer: COMPANY1)
```

g2-get-attributes-visible-in-mode

Returns the attributes that are accessible for a given user mode.

Synopsis

g2-get-attributes-visible-in-mode
(*class-or-item*: item-or-value, *user-mode*: symbol)
-> *visible-attributes*: sequence

Argument	Description
class-or-item	The item or class whose attributes you want to know about.
user-mode	The user mode in which attributes may be accessible.

Return Value	Description
visible-attributes	The names of the attributes that are accessible in a given user mode.

Description

This procedure returns a sequence of the attribute names that are accessible for an item in a given mode, after item and instance configurations are taken into account.

The first argument of this procedure is normally an item, but it can also be a symbol naming a class, in which case the returned sequence represents the accessible attributes for an instance of the class, even if the class is currently uninstantiated.

Example

This example employs the user-defined class equipment used in the example for g2-get-attribute-descriptions-of-class.

This example procedure call:

```
call g2-get-attribute-visible-in-mode(the symbol equipment,
                                      the symbol administrator)
```

returns a sequence that is shown here partially:

```
sequence(
    the symbol foundation-class,X
    the symbol class,
    the symbol uuid,
    . . . . . .
    the symbol producer)
```

g2-get-description-of-item

Gives you programmatic access to the describe menu choice of an item.

Synopsis

g2-get-description-of-item
(*item*: class item)
-> *description*: sequence

Argument	Description
item	The item you wish described.

Return Value	Description
description	A sequence of text values.

Example

This is an example of a sequence returned for a class-definition:

```
sequence("description of country.",
         "there are no methods defined specifically for this class.",
         "this is assigned to module definitions.")
```

g2-item-is-editable

Tells you whether an item is editable because its module can be saved to its current location or you have explicitly declared it editable.

Synopsis

g2-item-is-editable
(*item*: class item)
-> *return-value*: truth-value

Argument	Description
item	Any item in the current KB.

Return Value	Description
return-value	This procedure returns true when the item is editable; otherwise it returns false.

Description

An item is editable under any one of these conditions:

- It is a transient item.
- Its module file is writable.
- You have explicitly declared the item editable.

4

Color Change Operations

Describes procedures for controlling the colors of items on workspaces.

Introduction

The system procedures for changing colors let you access the different color regions of G2 icons, text boxes, and workspaces, such as an icon region, the background color of a workspace, or the text color of a message.

For more information about icons, items, and their color regions, see The Icon Editor and Icon Management in the *G2 Reference Manual*.

Accessing Color Regions and Color Patterns

All system-defined icons and items have named regions. You can get and set the colors of these regions by using the color change system procedures.

A **color pattern** is a symbol array containing the name of each color region of an item, such as the background color of a workspace. Each region name is followed by the name of the color or metacolor of that region. Thus, the elements of a color pattern are:

Element 0: region1,
Element 1: color1 or metacolor1,
Element 2: region2,
Element 3: color2 or metacolor2; and so on

Unnamed icon regions are not included in the color pattern.

Mode: Online HelpIDEach item and icon also has a color pattern that stores its currently displayed colors. In addition, each item or icon stores the following color patterns:

- A **permanent color pattern**, which is saved with the KB.

  If you change the color of one or more regions of an icon, you can make this change part of the permanent color pattern by invoking the make permanent action on the icon. If you subsequently save the KB, and then reset or reload it, the icon is displayed with the changes that you made to the permanent color pattern.

- A **default color pattern**, which stores the color pattern specified in the class definition of the item.

The colors stored in the permanent color pattern of an item are called the **permanent colors** of that icon. The colors stored in the default color pattern are called **default colors**.

The following items do not have color regions: graphs, trend charts, and user-defined icons whose class definitions do not specify color regions.

Note You cannot add or delete color regions by using the procedures described in this chapter. For more information, see The Icon Editor and Icon Management in the *G2 Reference Manual*.

---

g2-get-default-item-color

Returns the default color of an item region.

Synopsis

g2-get-default-item-color
(*item*: class item, *region*: symbol)
-> *default-color*: symbol

Argument	Description
item	An item.
region	The name of a color region of *item*.

Return Value	Description
default-color	The color or metacolor of *region*.

Description

G2-get-default-item-color returns the default color of the specified region of the item or icon, as defined in the item's definition or icon description.

Examples

The following user-defined procedure, get-default-color-pattern-using-get-color, creates a default color pattern for a specified item. The default color pattern is a symbol array named pattern.

To create the default color pattern, this procedure:

1. Calls g2-get-item-color-regions to read the names of the item's regions into a symbol array named regions-array.
2. Creates the symbol array pattern, and sets the length of this array to twice the length of the array regions-array. Pattern must be twice as long as regions-array because it contains both region names and colors.
3. Reads the region names in regions-array into the odd-numbered elements of pattern, starting with element 0.
4. Calls g2-get-default-item-color to get the default color of each region in regions-array, and reads these colors into the even-numbered elements of pattern.

```
get-default-color-pattern-using-get-color (item: class item)
     = (class symbol-array)
regions-array, pattern: class symbol-array;
i: integer;
pattern-length: integer;
color: symbol;

begin
     regions-array = call g2-get-item-color-regions (item);
     pattern-length = (the array-length of regions-array) * 2;
     create a symbol-array pattern;
     change the array-length of pattern to pattern-length;
     for i = 0 to (pattern-length -1) by 2
          do
               change pattern [ i ] = regions-array [round ( i/2 ) ];
               color = call g2-get-default-item-color (item, pattern[ i ]);
               change pattern [ (i + 1) ] = color;
          end;
     delete regions-array;
     return (pattern)
end
```

g2-get-default-item-color-pattern

Returns the default color pattern of an item.

Synopsis

g2-get-default-item-color-pattern
(*item*: class item)
-> *default-color-pattern*: class symbol-array

Argument	Description
item	The item whose color pattern is returned.

Return Value	Description
default-color-pattern	A symbol array containing a list of the color regions of *item*. Each color region is paired with the name of the color of that region.

Description

This procedure returns the default color pattern of an item.

Example

This example returns the default color pattern of an item named tulip to a symbol array named default-pattern.

```
tulip: class item;
default-pattern: class symbol-array;

default-pattern = call g2-get-default-item-color-pattern (tulip);
```

Returns the current color of a specified region of an item.

g2-get-item-color

Returns the current color of a specified region of an item.

Synopsis

g2-get-item-color
(*item*: class item, *region*: symbol)
-> *current-color*: symbol

Argument	Description
item	An item.
region	A region name of *item*.

Return Value	Description
current-color	A symbol value of the current color of *region*.

Description

This procedure returns the current color of the specified region of an item.

To return the permanent color of a region, use g2-get-permanent-item-color. To return the default color of a region, use g2-get-default-item-color.

Example

This example returns the color of a region named region-2 to a symbol named item-color.

```
item-color:symbol;
tulip: class item;
region-2:symbol;

begin

    item-color = call g2-get-item-color (tulip, region-2);

end
```

g2-get-item-color-pattern

Returns the current color pattern of a specified item.

Synopsis

g2-get-item-color-pattern
(*item*: class item)
-> *current-color-pattern*: class symbol-array

Argument	Description
item	The item whose color pattern is returned by this procedure.

Return Value	Description
current-color-pattern	A symbol array of the color pattern for *item*.

Description

G2-get-item-color-pattern returns the current color pattern of the specified item. The color pattern contains the names of the regions of the specified item. Each region name is paired with the name of the color of that region.

Example

This procedure, get-text-item-color-pattern, calls g2-get-item-color-pattern to read the current color pattern of an item into a symbol array named test-pattern.

```
get-test-item-color-pattern (item: class item) = (class symbol-array)
test-pattern: class symbol-array;
i: integer;

begin
     test-pattern = call g2-get-item-color-pattern (item);
     for i = 0 to ((the array-length of test-pattern) - 1) by 2
          do change test-pattern [i + 1] = test-color-array
               [round ( i/2 )];
          end;
     return (test-pattern)
end
```

The procedure then changes the color in the even-number elements of test-pattern to the colors in successive elements of an array named test-color-array. The value of element 0 of test-color-array is read into element 1 of test-pattern, the value of element 1 of test-color-array is read into element 3 of test-pattern, and so on.

g2-get-item-color-regions

Returns a symbol array containing the names of all the regions of a specified item.

Synopsis

g2-get-item-color-regions
(*item*: class item)
-> *region-names*: class symbol-array

Argument	Description
item	The item whose color regions are returned by this procedure.

Return Value	Description
region-names	A symbol array of the names of all regions of *item*. The names appear in the same order as in a symbol array returned by g2-get-item-color-pattern.

Description

G2-get-item-color-regions returns a symbol array of the names of all the color regions of the specified item.

Example

See the example for g2-get-default-item-color.

g2-get-permanent-item-color

Returns the color pattern of a specified item as it is when the KB is reset, or when the KB is saved and reloaded.

Synopsis

g2-get-permanent-item-color
(*item*: class item, *region*: symbol)
-> *permanent-color*: symbol

Argument	Description
item	An item.
region	A region of *item*.

Return Value	Description
permanent-color	A symbol of the color of *region*.

Example

This procedure, get-permanent-color-pattern-using-get-color, creates a permanent color pattern, named pattern, for an item.

```
get-permanent-color-pattern-using-get-color
    (item: class item) = (class symbol-array)
regions-array, pattern: class symbol-array;
i: integer;
pattern-length: integer;
color: symbol;

begin
    regions-array = call g2-get-item-color-regions (item);
    pattern-length = (the array-length of regions-array) * 2;
    create a symbol-array pattern;
    change the array-length of pattern to pattern-length;
    for i = 0 to (pattern-length -1) by 2
        do change pattern [ i ] = regions-array [round ( i/2 ) ];
            color = call g2-get-permanent-item-color (itm, pattern[ i ]);
            change pattern [ ( i + 1 ) ] = color;
        end;
    delete regions-array;
    return (pattern)
end
```

To create the permanent color pattern, this procedure:

1. Calls g2-get-item-color-regions to read the names of the item's regions into a symbol array named regions-array.
2. Creates the symbol array pattern, and sets the length of this array to twice the length of the array regions-array. Pattern must be twice as long as regions-array because it must contain both region names and the permanent color associated with each region.
3. Reads the region names in regions-array into the odd-numbered elements of pattern, starting with element 0.
4. Calls g2-get-permanent-item-color to get the permanent color of each region in regions-array, and reads these colors into the even-numbered elements of pattern.

g2-get-permanent-item-color-pattern

Returns the permanent color pattern of a specified item.

Synopsis

g2-get-permanent-item-color-pattern
(*item*: class item)
-> *permanent-color-pattern*: class symbol-array

Argument	Description
item	The item whose permanent color pattern is returned.

Return Value	Description
permanent-color-pattern	A symbol array of the permanent color pattern of *item*.

Example

An example that returns the permanent color pattern of an item named tulip to a symbol array named permanent-pattern, is:

```
tulip: item;
permanent-pattern: class symbol-array;

permanent-pattern = call g2-get-permanent-item-color-pattern (tulip)
```

g2-set-item-color

Sets the color of a specified color region of an item.

Synopsis

g2-set-item-color
(*item*: class item, *region*: symbol, *color*: symbol)

Argument	Description
item	An item.
region	The region of *item* whose color is set by this procedure.
color	The color to which *region* is set.

Example

An example that sets a color region is:

```
set-color-pattern-using-set-color
     (item: class item, pattern: class symbol-array)
i: integer;

begin
     for i = 0 to ((the array-length of pattern) - 1) by 2
          do call g2-set-item-color (item, pattern [ i ], pattern [ i + 1 ] )
          end
end
```

g2-set-item-color-pattern

Sets the color pattern of an item to the values in a specified symbol array.

Synopsis

g2-set-item-color-pattern
(*item*: class item, *pattern*: class symbol-array)

Argument	Description
item	An item.
pattern	A symbol array containing region names and color names.

Examples

This example sets the color pattern of an item named tulip to the color pattern specified by the symbol array named color-pattern:

```
tulip: class item;
color-pattern: class symbol-array;

call g2-set-item-color-pattern (tulip, color-pattern)
```

5

Connection Operations

Describes procedures that copy connections, and get information about them.

Introduction

Use the connection system procedures to copy the vertices of existing connections and to get information about connections between items.

Using Lists in Connection Procedures

The connection system procedures operate on lists that contain information about connections. These lists must exist at the time when the procedures are called. The procedures do not create the lists referenced by their arguments.

For information about how to create lists, see The Icon Editor and Icon Management in the *G2 Reference Manual*.

g2-get-connection-vertices

Gets the lengths of each segment of an existing connection and its corresponding vertices, and places these values in an integer-list.

Synopsis

g2-get-connection-vertices
(*item*: class item, *conn*: class connection,
item-list: class integer-list)

Argument	Description
item	The item with the connection.
conn	The connection.
item-list	An integer list that is populated by this procedure to contain the length of each connection segment, represented in workspace units.

Description

G2-get-connection-vertices gets the segments of an existing connection in the order they are drawn, beginning at the object. The segment connected to *item* is at element zero. When you specify the arguments for this procedure, *conn* must be connected to *item*. The integer list used for the connection segment lengths must exist when you execute the system procedure.

The purpose of g2-get-connection-vertices is specifically for copying an existing connection to reuse in creating another with the create action. The create action accepts an integer-list as an argument to create a new connection, using the grammar:

with the vertices given by *integer-list*

When populating an integer list with the connection segment lengths, g2-get-connection-vertices uses a positive number to specify a segment extending upwards or to the right of an object or vertex, and a negative number to specify a segment extending downwards or to the left.

The integer list that this procedure populates contains the minimum number of vertices that the create action requires. This number is typically one or two less than the total number of vertices in the original connection. The create action then infers the last one or two vertices based on the location of the item to which the newly created connection is being created.

Note The g2-get-connection-vertices system procedure has changed in recent G2 releases and currently returns the minimum number of vertices that the create action requires, rather than the exact number of vertices from the original connection. For KBs that may have relied on the previous behavior, a backward compatibility option exists to revert the system procedure to its previous behavior. For information about that option, see Procedures in the *G2 Reference Manual*.

Example

The following get-connection-vertices procedure illustrates one way to use the system procedure in conjunction with the create connection action. This procedure creates a new object with connections identical to those of an existing item, which is passed to the procedure as an argument.

```
get-connection-vertices(from-item: class object)
IL: class integer-list;
NewObject: class red-object;
C: class connection;
begin
    create an integer-list IL;
    change the name of IL to the symbol vertices-list;
    transfer vertices-list to this workspace at (50, -20);
    C = the connection connected to from-item;
    call g2-get-connection-vertices(from-item, C, vertices-list);
    create a red-object NewObject;
    transfer NewObject to this workspace at (50, 50);
    create a connection C of class connection connected to NewObjectl
        locating it at right 20 with the vertices given by vertices-list;
    make NewObject permanent
end
```

g2-get-items-connected-to-port

Finds items connected to the port of any given item, where the port name is given by an expression.

Synopsis

g2-get-items-connected-to-port
(*connection-source*: class item, *connected-class*: symbol, *port-name*: symbol, *connected-items*: class item-list)

Argument	Description
connection-source	The item with the port
connected-class	The class of the items that are connected to the first item (*connection-source*).
port-name	The name of the port on the *connection-source* item.
connected-items	The items connected to the port specified by *port-name*.

Description

G2-get-items-connected-to-port locates the items of a specified class (*connected-class*) that are connected to a specified item (*connection-source*) at a specified port (*port-name*). All items that are found are inserted at the end of a specified item-list (*connected-items*).

g2-get-items-connect-to-port does not clear *connected-items*. A single execution of g2-get-items-connect-to-port will not insert duplicates of any item into *connected-items*. However, repeated executions of g2-get-items-connect-to-port can insert duplicate items into the *connected-items* list, if the allow-duplicates attribute of that list is set to allow duplicates.

As with the any *class* connected expression, G2 does not signal an error if the given port is not defined.

You can refer to these connected items, using the expression any *class* connected at the *port* of *item*. However, in this expression, the port name is a symbol rather than an expression.

Example

The following call to g2-get-items-connected-to-port returns to c-list a list of circle items connected to the port named port1 on the item cc:

```
cc: class item;
c-list: class item-list;
c1, c2, c3: class circle;
s1, s2: class: square;

call g2-get-items-connected-to-port(cc, the symbol circle, the symbol port1, c-list);
```

If c1, c2, s1, s2 are connected to port1, and c3 is connected to port2, this call returns the following to c-list:

```
c1, c2
```

c1 and c2 are the only items of the class circle that are connected to port1.

Directory Operations

Describes procedures for accessing and changing the current directory.

Introduction

Use the directory operations system procedures to enable your G2 application to manipulate or obtain information about directories at the operating system level.

Using Wildcards with Directory Operations

You can use a wildcard in the text of the argument passed to the directory operation system procedures:

- g2-files-in-directory
- g2-subdirectories-in-directory

The procedures will return a list of names that meet the specified criteria.

Note The wildcard characters work on any G2 operating system from within G2.

For instance, you can use the string kb*s.kb to obtain a list of all KB files in a directory, whose file names begin with the characters `kb` and end with the characters `s.kb`.

To use wildcards, use combinations of the following characters:

Wildcard Character/ Purpose	Example
* (asterisk) Matches *zero or more* characters	Entering kb*s matches the files or directories named `kbfiles` and `kbs`.
? (question mark) Matches any one character	Entering kbfile? matches the files or directories named `kbfiles` and `kbfilez`
{*abc*} (braces) Matches *one* occurrence of the character *a* or *b* or *c*, where *a*, *b*, and *c* each represents a character	Entering kb{ef}iles matches the files or directories named `kbfiles` and `kbeiles`
{*abc*}* (braces and asterisk) Matches *zero or more* occurrences of the character *a* or *b* or *c*; where *a*, *b*, and *c* each represents a character	Entering kb{xyz}*files matches the files or directories named `kbfiles` and `kbzzzfiles`
! (exclamation point) Allows the use of other characters in the wildcard name	Entering kbfile!{s!} matches the file or directory named `kbfile{s}`

g2-change-default-directory

Changes the current default directory.

Synopsis

g2-change-default-directory
(*new-default*: text)

Argument	Description
new-default	The new default directory, in the format of the local operating system. The slash mark (/) after the final directory name in the path is optional, and can be omitted.

Example

This example (for a UNIX system) sets the default directory to `/home/dev/test-kbs`.

```
call g2-change-default-directory ("/home/dev/test-kbs/");
```

Note The slash mark (/) after test-kbs can be omitted.

g2-create-directory

Creates a new directory.

Synopsis

g2-create-directory
(*directory-name*: text, *create-parents*: truth-value)
-> *success*: truth-value

Argument	Description
directory-name	A text that provides the complete path name to the directory to create.
create-parents	If true, the procedure creates any directories in the path that do not exist, just like using mkdir -p on UNIX.

Return Value	Description
success	Returns true if the directory is created, and false otherwise.

g2-default-directory

Returns the name of the default working directory as a text string.

Synopsis

g2-default-directory
( )
-> *default-directory*: text

Return Value	Description
default-directory	The current default directory, in the format of the local operating system. In the case of UNIX directories, g2-default-directory returns the trailing slash mark (/) after the last directory name in the path.

Example

This example sets the default directory to `/home/dev/test-kbs` and returns the current default directory to `default-dir`.

```
default-dir: text;
. . .
call g2-change-default-directory("/home/dev/test-kbs/");
default-dir = call g2-default-directory( );
```

g2-devices-on-machine

Gets a list of all available devices on the machine from which you make the call.

Caution This procedure is for use on Windows platforms only.

Synopsis

g2-devices-on-machine
( )
-> *available-devices*: class text-list

Return Value	Description
available-devices	A text list of the names of available devices on the machine from which the call is made.

Description

This procedure returns a text-list of the names of all available devices on the machine where the call is made. On platforms other than Windows, this procedure always returns an empty text-list.

Example

Calling g2-devices-on-machine on your Windows machine might returns a text-list consisting of "C:", "D:", "E:", "H:", and "Z:". These are all legitimate Windows device names.

Note Windows device names are limited to one letter in the Roman alphabet, making the maximum possible number of elements in the text list 26.

g2-directory-exists

Tests whether a specified directory exists and returns true or false accordingly.

Synopsis

g2-directory-exists
(*directory-string*: text)
-> *directory-exists*: truth-value

Argument	Description
directory-string	The directory whose existence is tested, in the format of the local operating system.

Return Value	Description
directory-exists	true (the directory exists) or false (the directory does not exist).

Description

You can use g2-directory-exists to verify that a directory exists before you attempt to access the files within it.

Example

The following procedure tests whether a directory named `directory-to-access` exists:

```
test-directory-exists(directory-to-access: text)
exists-p: truth-value;

begin
    exists-p = call g2-directory-exists(directory-to-access);
    if exists-p then inform the operator for the next 30 seconds that
        "@"[directory-to-access]@" exists and is a directory"
    else inform the operator for the next 30 seconds that
        "@"[directory-to-access]@" does not exist or is not a directory"
end
```

The at sign (@) symbol preceding the quotation marks (") enables you to include quotation marks in messages.

g2-disk-space-available-in-directory

Returns the amount of disk space available in a specified directory.

Synopsis

g2-disk-space-available-in-directory
(*directory-string*: text)
-> *available-space*: integer

Argument	Description
directory-string	The directory for which the amount of available disk space is returned.

Return Value	Description
available-space	An integer of the amount of disk space available in the directory specified by *directory-string*. Disk space is shown in platform-specific units: kilobytes for UNIX and the number of free clusters on the disk for Windows.

Example

The following example returns to disk-space the available disk space for the directory `/home/dev/test-kbs`.

```
disk-space: integer;
. . .
disk-space = call g2-disk-space-available-in-directory("/home/dev/test-kbs");
```

Note The largest integer G2 supports is 536870911, therefore, this procedure will return 536870911 for disk space that exceeds that limit.

g2-files-in-directory

Gets a list of directory files.

Synopsis

g2-files-in-directory
(*directory-string*: text)
-> *files*: class text-list

Argument	Description
directory-string	The directory.

Return Value	Description
files	A text list containing the names of the files in the directory specified by *directory-string*.

Description

G2-files-in-directory returns a transient text-list of the files and subdirectories in the directory that you specify. Delete this list when you finish using it to avoid memory build-up.

Note Using Wildcards with Directory Operations describes the wildcard characters you can use with this system procedure to narrow the returned list.

Example

This example returns to files-in-dir a list of the files in `/home/dev/test-kbs`, and then prints the names of the files on the message board:

```
get-directory-files( )
files-in-dir: class text-list;
number-of-files, index: integer;
files-in-dir = call g2-files-in-directory("/home/dev/test-kbs");
index: integer;

begin
     for index = 0 to (number-of-files - 1)
          begin
               inform the operator that [files-in-dir[index]];
          end
     delete files-in-dir;
end
```

g2-subdirectories-in-directory

Gets a list of subdirectories.

Synopsis

g2-subdirectories-in-directory
(*directory-string*: text)
-> *subdirectories*: class text-list

Argument	Description
directory-string	The directory whose subdirectories are returned.

Return Value	Description
subdirectories	A text list containing the names of the subdirectories in the directory specified by *directory-string*.

Description

G2-subdirectories-in-directory returns a transient text-list of the subdirectories in the directory that you specify. Delete this list when you finish using it to avoid memory build-up.

Note Using Wildcards with Directory Operations describes the wildcard characters you can use with this system procedure to narrow the returned list.

Example

This example returns to subdirs-in-dir a list of the subdirectories in `/home/dev/test-kbs`.

```
subdirs-in-dir: class text-list;
. . .
subdirs-in-dir = call g2-subdirectories-in-directory("/home/dev/test-kbs");
```

7

Error and Message Handler Operations

Describes procedures for handling error messages, operator logbook messages, and message board messages within procedures and methods.

Introduction

Use these system procedures for handling and redirecting error messages and messages directed to the operator logbook or the message board.

Note More sophisticated techniques for managing messages are available through GFR. See the *G2 Foundation Resources User's Guide* for details.

g2-register-default-error-handler

Registers a default error handler to be used by the entire KB.

Synopsis

g2-register-default-error-handler
(*proc*: class procedure)

Argument	Description
proc	A procedure or method-declaration.

Description

This procedure registers the procedure or the methods associated with the method-declaration that you wish the entire KB to use as the default error handler. This handler shadows the system default handler. The procedure, or every method associated with the method-declaration, must take one argument, of class error.

Once registered, the default error handler remains in effect until either:

- The handler is deregistered, restoring the system default handler.
- A new handler is registered, superseding the existing one.

Resetting G2 does not affect handler shadowing: any handler registered remains in effect when G2 restarts.

Alternatives

More sophisticated techniques for managing error messages are available through GFR. See the *G2 Foundation Resources User's Guide* for details.

g2-deregister-default-error-handler

Deregisters a previously registered default error handler.

Synopsis

g2-deregister-default-error-handler
( )

Description

Use this procedure to deregister the currently registered default error handler. The system default handler is thereby unshadowed. If no handler was registered, the procedure does nothing.

Alternatives

More sophisticated techniques for managing error messages are available through GFR. See the *G2 Foundation Resources User's Guide* for details.

g2-get-default-error-handler

Returns the current default error handler.

Synopsis

g2-get-default-error-handler
( )
-> {*handler*: class procedure | false: truth-value}: item-or-value

Return Value	Description
handler	The default error handler that is currently registered.
false	Indicates that no default error handler is currently registered.

Description

Use this procedure to get the currently registered default error handler. G2 returns false if no handler is registered.

Alternatives

More sophisticated techniques for managing error messages are available through GFR. See the *G2 Foundation Resources User's Guide* for details.

g2-register-logbook-message-handler

Registers a logbook message handler.

Synopsis

g2-register-logbook-message-handler
(*proc*: class procedure)

Argument	Description
proc	The name of the procedure to use as the logbook message handler.

Description

This procedure registers the procedure that you wish the KB to use as the logbook message handler. This handler shadows the system-defined handler. The procedure registered must take one argument, of type text.

Once registered, the handler remains in effect until either:

- The handler is deregistered, restoring the system-defined handler.
- A new handler is registered, superseding the existing one.

Resetting G2 does not affect handler shadowing: any handler registered remains in effect when G2 restarts.

Alternatives

More sophisticated techniques for managing operator logbook messages are available through GFR. See the *G2 Foundation Resources User's Guide* for details.

g2-deregister-logbook-message-handler

Deregisters the previously registered logbook message handler.

Synopsis

g2-deregister-logbook-message-handler
( )

Description

Use this procedure to deregister the currently registered logbook message handler. The system-defined handler is thereby unshadowed. If no handler was registered, the procedure does nothing.

Alternatives

More sophisticated techniques for managing operator logbook messages are available through GFR. See the *G2 Foundation Resources User's Guide* for details.

g2-get-logbook-message-handler

Returns the currently registered logbook message handler.

Synopsis

g2-get-logbook-message-handler
( )
-> {*handler*: class procedure | false: truth-value}: item-or-value

Return Value	Description
handler	The logbook message handler that is currently registered.
false	Indicates that no logbook message handler is currently registered.

Description

Use this procedure to get the currently registered logbook message handler. G2 returns false if no handler is registered.

Alternatives

More sophisticated techniques for managing operator logbook messages are available through GFR. See the *G2 Foundation Resources User's Guide* for details.

g2-register-message-board-message-handler

Registers a message board handler.

Synopsis

g2-register-message-board-message-handler
(*proc*: class procedure)

Argument	Description
proc	The name of the procedure to use as the message board handler.

Description

This procedure registers the procedure to use as the handler for all message board messages. This handler shadows the system-defined handler. The procedure registered must take one argument, of type text.

Once registered, the handler remains in effect until either:

- The handler is deregistered, restoring the system-defined handler.
- A new handler is registered, superseding the existing one.

Resetting G2 does not affect handler shadowing: any handler registered remains in effect when G2 restarts.

Alternatives

More sophisticated techniques for managing message book messages are available through GFR. See the *G2 Foundation Resources User's Guide* for details.

g2-deregister-message-board-message-handler

Deregisters the currently registered message board message handler.

Synopsis

g2-deregister-message-board-messsge-handler
( )

Description

Use this procedure to deregister the currently registered message board message handler. The system-defined handler is thereby unshadowed. If no handler was registered, the procedure does nothing.

Alternatives

More sophisticated techniques for managing message book messages are available through GFR. See the *G2 Foundation Resources User's Guide* for details.

g2-get-message-board-message-handler

Returns the currently registered message-board handler.

Synopsis

g2-get-message-board-message-handler
( )
-> {*handler*: class procedure | false: truth-value}: item-or-value

Return Value	Description
handler	The message-board handler that is currently registered.
false	Indicates that no message-board handler is currently registered.

Description

Use this procedure to get the currently registered message board handler. G2 returns false if no handler is registered.

Alternatives

More sophisticated techniques for managing message book messages are available through GFR. See the *G2 Foundation Resources User's Guide* for details.

8

File Operations

Describes procedures that provide access to and management of files external to a KB.

Introduction

Use the file operation system procedures for:

- Accessing files
- Obtaining file characteristics
- Manipulating filestrings
- Reading and writing files

The file I/O system procedures write KB data to an external file, and read data from an external file into a KB.

Note File operation system procedures that accept file names as an argument accept the syntax of the operating system on which G2 is running.

Using Lists in File Procedures

Some file system procedures return transient lists of directory information. These lists are created by the procedures themselves; you do not have to create the lists that receive return values.

However, file procedures do not create the lists referenced by their arguments. These lists must exist at the time when the procedures are called. For information about how to create lists, see Lists and Arrays in the *G2 Reference Manual*.

Using G2-Stream Objects in File Procedures

When you perform file I/O operations, the system procedures may return a truth-value, indicating success or failure of the operation, or a g2-stream object, through which you can reference the open file.

Instances of g2-stream objects are created and returned when G2 opens a file using these system procedures:

- g2-open-file-for-append
- g2-open-file-for-read
- g2-open-file-for-read-and-write
- g2-open-file-for-write

If a G2-stream object is transferred to a workspace, the object appears as:

Once a G2-stream exists, you can query its status by referring to its g2-stream-status attribute. Further, you may need to change the value of the text-conversion-style attribute as described in Using Text Conversion Styles.

These system procedures delete a G2-stream object and close one or more open files:

- g2-close-all-files
- g2-close-file

Filtering Escape Characters

File operation system procedures that accept text arguments and use the Gensym character set filter embedded G2 escape characters appropriately before passing

the text to the operating system. For more information about escape characters, see G2 Character Support in the *G2 Reference Manual*.

G2 Stream Attributes

A G2-stream has these attributes.

Attribute	Description
name-of-file	The text string that you used to open the file.
Allowable values:	Any valid file name
Default value:	none
Notes:	No default value exists for path name components, such as version or device.
file-system	A symbol naming the operating system in which you opened the stream.
Allowable values:	unix, dos, and win32
Default value:	none
access-direction	A symbol describing the access direction.
Allowable values:	inactive, input, output, and input-and-output
Default value:	none
position-in-file	An integer indicating the file position, which is measured in characters. The value of this attribute may also be a symbol such as end-of-file-reached.
Allowable values:	integer | symbol
Default value:	none

Attribute	Description
g2-stream-status	A symbol providing the status of the last operation involving this stream.
Allowable values:	Listed in applicable file operations system procedures.
Default value:	none
text-conversion-style	The name of the text-conversion-style item to use when performing file operations requiring specific character encoding. Text conversion style items and character encoding are described in Text Parsing and Manipulation and G2 Character Support in the *G2 Reference Manual*. Changing the value of the text-conversion-style attribute when reading from or writing to a file causes G2 to signal an error.
Allowable values:	A symbol providing the name of the text-conversion-style item to use. When no specific style is specified, G2 uses an internal default style, specifying the Gensym character set.
Default value:	none

Using Text Conversion Styles

During file operations, G2 uses the text conversion style specified in the text-conversion-style attribute of the g2-stream object. When G2 creates a new g2-stream object, no default text conversion style is specified. If you do not specify a style, G2 uses the internal default style. We recommend that you create your own text-conversion-style to represent the character sets you require. For example, you could create three text-conversion-styles to handle file I/O for each of these character sets:

- ASCII
- Korean
- Cyrillic

To use a pre-existing text-conversion-style:

➔ Change the value of the text-conversion-style attribute to the appropriate style as soon as the g2-stream object exists after a file open operation, and before performing any file I/O operations.

To change the value of the text-conversion-style attribute programmatically:

➔ Use a conclude statement such as the one shown next in part of a file I/O procedure.

```
file-open(filename: text)
Stream: class g2-stream;
Line: text;
Filetext: text = "";
Exist: truth-value;
begin
     Exist = call g2-file-exists(filename);
     if not (Exist)
          then post "[filename] does not exist."
          else
               begin
                    Stream = call g2-open-file-for-read(filename);
                    transfer Stream to this workspace at (50, 50);
                    conclude that the text conversion-style of Stream
                         = the symbol ascii-style;
     . . . . . .
```

Handling Special Characters

Failure to change the text-conversion-style attribute to the appropriate style can result in characters being misinterpreted. For example, if the default gensym text-conversion-style is in use, and you are reading from an ASCII text file, G2 assumes the characters you are importing are in the Gensym character set. In the Gensym character set, you must use the tilde (~) escape character to represent a backslash (\) character literally, as in a path separator within a string such as:

```
c:\mydir\newfile.txt
```

If you do not encode backslash characters correctly in the import file, G2 interprets each backslash (\) as an escape sequence itself.

To have G2 interpret a backslash (\) character literally in an ASCII import file:

1. Create a text-conversion-style that specifies us-ascii as the external character set.
2. Change the text-conversion-style attribute of the g2-stream object to that style before starting file I/O operations.

For information about encoding special characters for file I/O purposes and using text-conversion-style items, see Text Parsing and Manipulation and G2 Character Support in the *G2 Reference Manual*.

Using File Operations

The following example shows you how to use file-related procedures to read a file and display its results on a workspace.

Company X-Supplies produces screws. There are four series of drywall screws corresponding to four materials that are used to make them. The management at Company X-Supplies wants you to configure G2 to display the quality assurance results for all drywall screw lengths within a certain series. The quality assurance results are contained in a text file and updated automatically at the end of each day.

To read the quality assurance text file for management to review the results:

1 Create a procedure that reads your file and displays the results as a message on a workspace. For example:

```
open-file-for-text-display(filename: text)
text-file-stream: class g2-stream;
each-line: text;
all-lines: text = "";
counter: integer = 0;
begin
     text-file-stream = call g2-open-file-for-read(filename);
     repeat
          exit if the g2-stream-status of text-file-stream =
               the symbol end-of-file-reached;
          each-line = call g2-read-line(text-file-stream);
          all-lines = "[all-lines] [each-line]";
          post "[counter]: [each-line] is being read.";
          counter = counter + 1
     end;
     change the text of text-display-message to all-lines;
     call g2-close-file(text-file-stream)
end
```

Note that this procedure opens a g2-stream for reading and then closes it.

2 Create an action button to start the open-file-for-text-display procedure with the quality assurance file name (qa-results.text) as an argument. The action is:

```
start open-file-for-text-display ("/path/path/qa-results.text")
```

3 Create a message and give it the name text-display-message.

When you click the Display QA Results button, G2 opens a g2-stream, reads the `qa-results.text` file (informing the operator as it progresses) and then fills the text-display-message with the contents of the file.

Accessing Files

These system procedures let you access files:

g2-close-all-files

g2-close-file

g2-delete-file

g2-open-file-for-append

g2-open-file-for-read

g2-open-file-for-read-and-write

g2-open-file-for-write

g2-rename-file

g2-set-file-position

g2-close-all-files

Closes all G2 streams and then calls g2-close-file to close all of the files associated with those streams.

Synopsis

g2-close-all-files
    ( )

g2-close-file

Deletes a specified g2-stream item and closes the file with which that stream was associated.

Synopsis

g2-close-file
(*stream-to-close*: class g2-stream)

Argument	Description
stream-to-close	The G2 stream to delete before closing the file.

Example

The following example deletes the G2 stream stream-1 and closes the file associated with that stream.

```
stream-1: class g2-stream;
. . .
call g2-close-file (stream-1);
```

g2-delete-file

Deletes the specified file. You cannot call g2-delete-file when there is a g2-stream item open to this file or when some other process has opened the file.

Synopsis

g2-delete-file
(*filestring*: text)

Argument	Description
filestring	A text string specifying the file to delete.

Example

The following example deletes the file `my-testkb.kb`.

```
call g2-delete-file ("/home/test-kbs/my-testkb.kb");
```

g2-open-file-for-append

Opens a file for appending, optionally creating a g2-stream for this purpose if one does not already exist.

Synopsis

g2-open-file-for-append
(*filestring*: text, *create-it*: truth-value)
-> *appending-stream*: class g2-stream

Argument	Description
filestring	The name of the file to open for appending to.
create-it	If true, creates a G2 stream for appending to the file specified by *filestring*, if a G2 stream does not already exist. If false, does not create a G2 stream. If no stream exists, signals an error.

Return Value	Description
appending-stream	The G2 stream created for appending to the specified file.

Description

G2-open-file-for-append creates a G2 stream (if *create-it* is set to true) for appending to the file with the name specified by *filestring*. G2 signals an error if the file cannot be opened.

G2-stream-status

open-for-append

Example

This example creates a stream for appending to the file `file-to-append.txt`, opens that file, writes two bytes to the file, and closes the file.

```
file: class g2-stream;
...
file = call g2-open-file-for-append ("file-to-append.txt", true);
call g2-write-byte (file, 98);
call g2-write-byte (file, 99);
call g2-close-file (file);
```

g2-open-file-for-read

Creates a g2-stream item with the given name for reading from a file and opens the file for reading.

Synopsis

g2-open-file-for-read
(*filestring*: text)
-> *read-stream*: class g2-stream

Argument	Description
filestring	The name of the file to open for reading.

Return Value	Description
read-stream	The G2 stream created for reading from the specified file.

Description

G2-open-file-for-read creates a G2 stream with the given name for reading from a file. G2 signals an error if the file cannot be opened, or if an attempt is made to write to the file.

G2-stream-status

open-for-read

Example

The following example creates a G2 stream for reading from the file `file-to-append.txt`, opens the file for reading, sets a file position in the file, and reads two bytes from it:

```
file: class g2-stream;
...
file = call g2-open-file-for-read ("file-to-append.txt");
call g2-set-file-position (file, 1, false);
byte-read-1 = call g2-read-byte (file);
byte-read-2 = call g2-read-byte (file);
```

g2-open-file-for-read-and-write

Creates a g2-stream item for both reading and writing to a file.

Synopsis

g2-open-file-for-read-and-write
(*filestring*: text)
-> *read-write-stream*: class g2-stream

Argument	Description
filestring	The name of the file to open for reading and writing.

Return Value	Description
read-write-stream	The G2 stream created for reading from and writing to the specified file.

Description

G2-open-file-for-read-and-write creates a G2 stream for both reading and writing to a file. If your operating system does not support bidirectionality, G2 signals an error. G2 signals an error if the file cannot be opened.

G2-stream-status

open-for-read-and-write

Example

The following example opens the file `readwrite.txt` for reading and writing and writes a byte to it:

```
file-rw: class g2-stream;
...
file-rw = call g2-open-file-for-read-and-write ("readwrite.txt");
call g2-write-byte (file-rw, 99);
```

g2-open-file-for-write

Creates a g2-stream item for writing to a specified file and then opens the file.

Synopsis

g2-open-file-for-write
(*filestring*: text)
-> *write-stream*: class g2-stream

Argument	Description
filestring	The name of the file to open for writing.

Return Value	Description
write-stream	The G2 stream created for writing to the specified file.

Description

G2-open-file-for-write creates a G2 stream with the given name for writing to a file. G2 signals an error if it cannot open the file.

G2-stream-status

open-for-write
tried-read-byte-on-stream-not-opened-for-read
tried-read-byte-on-line-not-opened-for-read

The last two status values occur if a read is attempted on the stream.

Example

The following example creates a G2 stream for writing to the file `readwrite.txt`, opens that file, and writes a byte to it:

```
file-rw: class g2-stream;
...
file-rw = call g2-open-file-for-write ("readwrite.txt");
call g2-write-byte (file-rw, 99);
```

g2-rename-file

Gives a new name to a file.

Synopsis

g2-rename-file
(*old-filestring*: text, *new-filestring*: text)

Argument	Description
old-filestring	The current file name.
new-filestring	The new file name, which replaces *old-filestring*.

Description

G2-rename-file gives a new name to a file.

You cannot call g2-rename-file when there is a G2 stream open to the file or when another process has opened the file.

Example

```
call g2-rename-file ("my-file.txt", "new-my-file.txt");
```

g2-set-file-position

Changes the G2 stream to a specified file position.

Synopsis

g2-set-file-position
(*stream*: class g2-stream, *new-position*: integer,
extend-file-if-necessary: truth-value)

Argument	Description
stream	The G2 stream whose file position is updated.
new-position	The new file position, expressed as the number of bytes (characters) from the beginning of the file.
extend-file-if-necessary	If true, the file is extended if the *new-position* value is beyond the end of the file. If false, the file is not extended if the *new-position* value is beyond the end of the file.

Description

G2-set-file-position changes the G2 stream to the file position specified by *new-position*. If the host operating system does not support this, it signals an error. If the *new-position* goes beyond the end of file marker, and *extend-file-if-necessary* is false, an error occurs.

G2-stream-status

successfully-updated-position

Example

This example opens the file `readwrite.txt` for reading, sets the G2 stream to a new file position in this file, and reads one byte from the file:

```
file: class g2-stream;
byte-read-1: integer;
...
file = call g2-open-file-for-read ("readwrite.txt");
call g2-set-file-position (file, 1, false);
byte-read-1 = call g2-read-byte (file);
```

Obtaining File Characteristics

G2 provides these system procedures for obtaining file characteristics:

- g2-file-exists
- g2-file-names-are-identical
- g2-latest-date-file-attributes-were-changed
- g2-latest-date-file-was-accessed
- g2-latest-date-file-was-modified
- g2-length-of-file
- g2-type-of-file-system

g2-file-exists

Tests whether or not a file exists and returns true or false accordingly.

Synopsis

g2-file-exists
(*filestring*: text)
-> *file-exists*: truth-value

Argument	Description
filestring	The name of the file whose existence is to be tested.

Return Value	Description
file-exists	The value true if the file specified by *filestring* exists, or false if it does not.

Example

This example returns true or false to file-exists, depending on whether the file `my-file.txt` exists.

```
file-exists: truth-value;
...
file-exists = call g2-file-exists ("my-file.txt");
```

If the parameter unicode-for-filenames? is setted to yes it would be possible to use it with Unicode characters:

```
file-exists = call g2-file-exists ("c:\日本.txt")
```

g2-file-names-are-identical

Compares two file names and returns true if they are identical.

Synopsis

g2-file-names-are-identical
(*filestring1*: text, *filestring2*: text)
-> *identical-names*: truth-value

Argument	Description
filestring1	The name of the first file.
filestring2	The name of the second file.

Return Value	Description
identical-names	The value true if the file names are identical, or false if the names differ.

Example

This example returns false to identical because the file names are different:

```
identical: truth-value;
...
identical = call g2-file-names-are-identical ("file-one.txt", "file-two.txt");
```

g2-latest-date-file-attributes-were-changed

Returns the time when the file attributes were most recently opened, written, renamed, or had their permissions changed.

Synopsis

g2-latest-date-file-attributes-were-changed
(*filestring*: text)
-> *time-in-seconds*: float

Argument	Description
filestring	The name of the file.

Return Value	Description
time-in-seconds	A float representing the number of seconds indicating when the file attributes were most recently changed. This time is expressed as the number of seconds that had elapsed since January 1, 1970 Greenwich Mean Time, at the moment when the file attributes were changed.

Description

File attributes are platform-dependent but typically include access permissions and modification dates. G2 signals an error if it cannot determine when the attributes were most recently changed.

Example

This example returns to change-time the time when attributes of the file were changed most recently.

```
change-time: float;
...
change-time = call g2-latest-date-file-attributes-were-changed ("my-file.txt");
```

g2-latest-date-file-was-accessed

Returns the time when the file was most recently opened, read, or written.

Synopsis

g2-latest-date-file-was-accessed
(*filestring*: text)
-> *time-in-seconds*: float

Argument	Description
filestring	The name of the file.

Return Value	Description
time-in-seconds	A float value of the number of seconds indicating when the file was most recently accessed. This time is expressed as the number of seconds that had elapsed since January 1, 1970 Greenwich Mean Time, at the moment when the file was most recently accessed.

Description

G2 signals an error if it cannot determine when the file was most recently accessed. Note that what constitutes file access is platform-dependent.

Example

This example returns to access-time the time when the file `my-file.txt` was accessed most recently.

```
access-time: float;
...
access-time = call g2-latest-date-file-was-accessed ("my-file.txt");
```

g2-latest-date-file-was-modified

Returns the time when the file was most recently modified.

Synopsis

g2-latest-date-file-was-modified
(*filestring*: text)
-> *time-in-seconds*: float

Argument	Description
filestring	The name of the file.

Return Value	Description
time-in-seconds	A float value of the number of seconds indicating when the file was most recently modified. This time is expressed as the number of seconds that had elapsed since January 1, 1970 Greenwich Mean Time, at the moment when the file was most recently modified.

Description

G2 signals an error if it cannot determine when the file was most recently modified. Note that what constitutes file modification is platform-dependent, but typically includes when the file was opened for writing or appending.

Example

This example returns to modify-time the time when the file `my-file.txt` was modified most recently.

```
modify-time: float;
...
modify-time = call g2-latest-date-file-was-modified ("my-file.txt");
```

g2-length-of-file

Returns the length (in bytes) of the file associated with an open G2 stream.

Synopsis

g2-length-of-file
(*file-stream*: class g2-stream)
-> *file-length*: integer

Argument	Description
file-stream	The G2 stream.

Return Value	Description
file-length	An integer of the length of the file, in bytes.

Description

G2-length-of-file returns the length (in bytes) of the file associated with an open G2 stream. If G2 cannot determine the file length, it signals an error.

G2-stream-status

successfully-obtained-file-length

Example

This example returns to length the length of the file associated with the G2 stream stream-1.

```
stream-1: class g2-stream;
length: integer;
...
length = call g2-length-of-file (stream-1);
```

g2-type-of-file-system

Returns the type of the native operating system.

Synopsis

g2-type-of-file-system
( )
-> *operating-system*: symbol

Return Value	Description
operating-system	A symbol indicating the operating system type. Possible values are: unix, dos, and win32.

Example

This example returns to file-system the type of the native operating system.

```
file-system: symbol;
. . .
file-system = call g2-type-of-file-system( );
```

Manipulating Filestrings

G2 provides system procedures that enable you to:

- Create filestrings
- Obtain filestring components
- Convert filestrings to lists
- Convert a list of directory path names into a string of text.

These system procedures are:

g2-collect-into-filestring

g2-file-base-name-string

g2-file-device-string

g2-file-directory-list-to-string

g2-file-directory-string

g2-file-directory-string-to-list

g2-file-extension-string

g2-file-host-string

g2-file-version-string

g2-partition-filestring

g2-collect-into-filestring

Creates a filestring with its arguments.

Synopsis

g2-collect-into-filestring
(*host*: text, *device*: text, *directory*: text, *root-name*: text,
extension: text, *version*: text)
-> *filestring*: text

Argument	Description
host	The host name of the computer that stores the file.
device	(Windows only) The device that stores the file.
directory	The directory that contains the file.
root-name	The file name, with extension or version indicator.
extension	The file extension.
version	The version number or indicator.

Return Value	Description
filestring	A text string of the arguments given to this system procedure.

Description

G2-collect-into-filestring creates a filestring with its arguments. To test whether the filestring exists, use the g2-file-exists system procedure.

This procedure performs the reverse of the operation performed by g2-partition-filestring. Some of the returned fields are not applicable for all systems. An empty string ("") is returned for those fields.

Example

The following example returns a filestring to collect-path-string.

```
collect-path-string, host, device, directory, root-name, extension,version: text;
. . .
collect-path-string = call g2-collect-into-filestring (host, device, directory,
     root-name, extension, version);
```

g2-file-base-name-string

Extracts the text specifying the file name from the given text.

Synopsis

g2-file-base-name-string
(*file-string*: text)
-> *base-name*: text

Argument	Description
file-string	The complete filestring from which the base file name is extracted.

Return Value	Description
base-name	A text string of the base file name from *file-string*.

Description

G2-file-base-name-string extracts the text specifying the file name (without the extension or version, if any) from the given text.

Example

The following procedure returns "my-test-application" to base-file-name.

```
base-file-name: text;
...
base-file-name = call g2-file-base-name-string
    ("/home/dev/test-kbs/my-test-application.kb");
```

g2-file-device-string

Extracts the text specifying the device from the given text.

Synopsis

g2-file-device-string
(*file-string*: text)
-> *file-device*: text

Argument	Description
file-string	The complete filestring from which the device is extracted.

Return Value	Description
file-device	A text string of the device, extracted from *file-string*. Only certain operating systems, such as Windows, include a device specification in files strings.

Description

G2-file-device-string extracts the text specifying the device from the given text. Note that in UNIX systems, g2-file-device-string always returns an empty string (""), not the device mount point.

Example

The following example returns my$device to dev-string:

```
dev-string: text;
file-string: text = "my$device:[mydir.subdir]myfile.type;2";
. . .
dev-string = call g2-file-device-string (file-string);
```

g2-file-directory-list-to-string

Converts a text-list of directory pathnames into a text string.

Synopsis

g2-file-directory-list-to-string
(*directory-list*: class text-list)
-> *names-to-path*: text

Argument	Description
directory-list	The directory list to be converted into a text string.

Return Value	Description
names-to-path	The text string containing a directory pathname.

Description

G2-file-directory-list-to-string converts a text-list of directory pathnames into a text string. This procedure performs the reverse of the operation performed by g2-file-directory-string-to-list.

For example, g2-file-directory-list-to-string converts the following text list:

```
"home", "dev", "test-kbs"
```

into the following text string:

```
"/home/dev/test-kbs/"
```

Example

The following example converts a pathname string to a text-list and then reconverts the text-list to a pathname string:

```
directory-string: text;
directory-list: class text-list;
...
directory-list = call g2-file-directory-string-to-list ("/home/dev/test-kbs/");
directory-string = call g2-file-directory-list-to-string (directory-list);
```

g2-file-directory-string

Extracts the text specifying the directory from the given text.

Synopsis

g2-file-directory-string
(*file-string*: text)
-> *file-directory*: text

Argument	Description
file-string	The filestring from which the directory specification is extracted.

Return Value	Description
file-directory	A text string containing the directory extracted from *file-string*.

Description

G2-file-directory-string extracts the text specifying the directory from the given text.

Example

This example returns "/home/test-kbs" to directory-string:

```
directory-string: test;
full-path: text = "/home/test-kbs/my-app.kb";
...
directory-string = call g2-file-directory-string (full-path);
```

g2-file-directory-string-to-list

Converts a text string into a test-list item.

Synopsis

g2-file-directory-string-to-list
(*file-string*: text)
-> *path-to-names*: class text-list

Argument	Description
file-string	The file directory string to be converted into a text list.

Return Value	Description
path-to-names	A transient text-list containing the individual parts of *file-string*.

Description

G2-file-directory-string-to-list converts a text string representing a directory pathname into a text-list of its components as text. This procedure performs the reverse of the operation performed by g2-file-directory-list-to-string.

For example, g2-file-directory-string-to-list converts the following text string:

"/home/dev/test-kbs/"

into the following text list:

"home", "dev", "test-kbs"

Return Value	Description
class text-list	The text list containing the components of the text directory string.

Example

See the example for g2-file-directory-list-to-string.

g2-file-extension-string

Extracts the text specifying the file type from the given text.

Synopsis

g2-file-extension-string
(*file-string*: text)
-> *extension*: text

Argument	Description
file-string	The complete filestring from which the file extension is extracted.

Return Value	Description
extension	A text string of the file extension extracted from *file-string*.

Description

G2-file-extension-string extracts the text specifying the file type from the given text.

Example

The following example returns ".kb" to file-extension:

```
file-extension: text;
...
file-extension = call g2-file-extension-string
    ("/home/dev/test-kbs/my-test-application.kb");
```

g2-file-host-string

Extracts the text specifying the host name from a specified text string.

Synopsis

g2-file-host-string
(*file-string*: text)
-> *host-name*: text

Argument	Description
file-string	The filestring from which the host specification is extracted.

Return Value	Description
host-name	A text string of the host specification extracted from *file-string*.

Description

G2-file-host-string extracts the text specifying the file host from the given text. If the host name is not applicable to the file system operations, an empty string is returned.

Example

```
host, path-string: text;
...
host = call g2-file-host-string (path-string);
```

g2-file-version-string

Extracts the text specifying the file version from the given text.

Synopsis

g2-file-version-string
(*file-string*: text)
-> *file-version*: text

Argument	Description
file-string	The filestring from which the file version is extracted.

Return Value	Description
file-version	A text string of the file version extracted from *file-string*.

Description

G2-file-version-string extracts the text specifying the file version from the given text. If there is no explicit version, it returns an empty string ("").

Example

In the following example (for Unix), g2-file-version-string returns a tilde (~) to version:

```
version : text;
path-string: text = "/home/dev/test-kbs/my-application.kb~";
...
version= call g2-file-version-string (path-string);
```

g2-partition-filestring

Accepts a filestring and returns the host, device, directory, base-name, extension, and version of that file.

Synopsis

g2-partition-filestring
(*file-string*: text)
-> *host*: text, *device*: text, *directory*: text, *base-name*: text, *extension*: text, *version*: text

Argument	Description
file-string	The filestring that is to be separated into its components.

Return Value	Description
text	The directory.
text	The base-name.
text	The extension.
text	The version.
empty string ("")	Returned if *file-string* is not applicable to the file system.

Description

G2-partition-filestring accepts a filestring and returns the host, device, directory, base-name, extension, and version of that file. This procedure performs the reverse of the operation performed by g2-collect-into-filestring.

Example

```
path-string: text = "/home/dev/test-kbs/my-application.kb~";
host, dev, dir, b-name, ext, ver: text;
...
host, dev, dir, b-name, ext, ver = call g2-partition-filestring (path-string);
```

Reading and Writing Files

The system procedures for reading and writing files enable you to read from and write to files outside of G2, at the operating system level.

Note For these system procedures, a byte is a value from 0 - 255. The procedures are limited to the 1,000,000 byte length enforced by Gensym strings.

You can also use these procedures to represent floating-point numbers exactly as they appear in your KB.

In general, the system procedures that perform I/O through streams uses the same procedure names as the system procedures that perform I/O using sockets. However, note that the I/O system procedures for both streams and sockets are both implemented as methods rather than as procedures. See Network Reading and Writing.

- g2-bytes-to-float
- g2-float-to-bytes
- g2-float-to-text
- g2-stream::g2-read-byte
- g2-stream::g2-read-bytes-as-sequence
- g2-stream::g2-read-bytes-as-text
- g2-stream::g2-read-line
- g2-stream::g2-write-byte
- g2-stream::g2-write-bytes
- g2-stream::g2-write-line
- g2-write-line-in-gensym-charset
- g2-stream::g2-write-string
- g2-write-line-in-gensym-charset
- g2-bytes-to-float

g2-bytes-to-float

Decodes the floating-point number that was generated in your KB.

Synopsis

g2-bytes-to-float
(*byte1*: integer, *byte2*: integer, *byte3*: integer, *byte4*: integer,
byte5: integer, *byte6*: integer, *byte7*: integer, *byte8*: integer)
-> *float-value*: float

Argument	Description
byte1, byte2, ... byte8	The integer bytes used to store the floating point value in a platform-independent format.

Return Value	Description
float-value	The floating point value stored as integer bytes.

Description

G2-bytes-to-float decodes, in a platform-independent way, the exact floating-point number that was generated in your KB. Because different platforms have different formats for representing floating-point values, you may find it useful to store floating point values as integer bytes.

This procedure performs the reverse of the operation that g2-float-to-bytes performs.

Example

This example converts integer bytes to a floating-point value and returns this value to float-value:

```
float-value: float;
b1, b2, b3, b4, b5, b6, b7, b8: integer;
...
float-value = call g2-bytes-to-float(b1, b2, b3, b4, b5, b6, b7, b8);
```

g2-float-to-bytes

Encodes the floating-point number that was generated in your KB.

Synopsis

g2-float-to-bytes
(*this-float*: float)
-> *byte1*: integer, *byte2*: integer, *byte3*: integer, *byte4*: integer,
byte5: integer, *byte6*: integer, *byte7*: integer, *byte8*: integer

Argument	Description
this-float	The floating-point value that is to be encoded in bytes.

Return Value	Description
byte1 [, ...]	The integer bytes used to store the floating point value in a platform-independent format.

Description

G2-float-to-bytes encodes, in a platform-independent way, the exact floating-point number that was generated in your knowledge base. This procedure is useful for storing the exact floating-point value in a file.

This procedure performs the reverse of the operation performed by the g2-bytes-to-float procedure.

Example

This example converts the floating-point value float-value into a series of integer bytes:

```
float-value: float = 10.0;
b1, b2, b3, b4, b5, b6, b7, b8: integer;
...
b1, b2, b3, b4, b5, b6, b7, b8 = call g2-float-to-bytes(float-value);
```

g2-float-to-text

Formats the display of a float value within a text string.

Synopsis

g2-float-to-text
(*float-value*: float, *minimum-width*: integer, *precision*: integer,
output-format: symbol, *remove-trailing-zeros-after-decimal*: truth-value)
-> *formatted-float-in-text*: text

Argument	Description
float-value	The float value to format as text.
minimum-width	Specifies the minimum text width. This number represents the total number of characters in the text. If the formatted float value has fewer characters than the number you specify, the text is padded on the left side with space characters to equal the given width. If the formatted value has more characters than the specified width, the width expands to accommodate the text without adding any space characters.
precision	Specifies either the number of digits to the right of the decimal point, or the significant digits, depending on the output-format value. When the formatted float uses the float or exponent output-format, precision indicates the digits to the right of the decimal point. When using the best output-format, precision determines the number of significant digits.

Argument	Description
output-format	Determines the precision format. Valid formats are: • float, which displays the value in a float format • exponent, which displays the value as an exponent • best, which displays the value as a float or an exponent if the result is either too small or too large for the specified precision. • force-zero, similar with best, except that the result is accurate and using minimal amount of mantissa digits to represent the value. There's at least one digit after the decimal point, even it's zero. *precision* is ignored in this case.
remove-trailing-zeros-after-decimal	Specifies whether to remove trailing zeros to the right of the decimal point. A value of true removes trailing zeros; a value of false does not.

Note When you use the best output-format, trailing zeros are always removed from the right of the decimal, regardless of how you specify the remove-trailing-zeros-after-decimal argument.

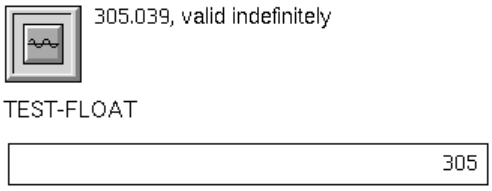
Return Value	Description
formatted-float-in-text	A text string of the formatted float value.

Description

Formats a float value to a right-justified text string, demonstrated in the example.

Example

Given a float variable, test-float, the next procedure displays the formatted float value in a message, using the output-format best:




```
test-format(float-value: float, msg: class message)
string: text;
begin
     string = call g2-float-to-text(float-value, 50, 4, the symbol best, false);
     change the text of msg to string
end
```

g2-stream::g2-read-byte

Reads a byte from a G2 stream and returns it either as an integer or as -1 if G2 is unable to read it.

Synopsis

g2-stream::g2-read-byte
(*stream*: class g2-stream)
-> *stream-byte*: integer

Argument	Description
stream	The G2 stream through which a byte is read.

Return Value	Description
stream-byte	An integer byte read from *stream*.

Description

G2-read-byte reads a byte from a G2 stream and returns it either as an integer or as -1 if G2 is unable to read it.

You can perform random-access reads by using g2-read-byte with the g2-set-file-position procedure if your operating system supports this.

G2-stream-status

successfully-read-byte
error-during-read-byte
end-of-file-reached

Example

The following reads a byte from the file `readwrite.txt` and returns the byte to byte-read-1:

```
file: class g2-stream;
byte-read-1:integer;
...
file = call g2-open-file-for-read ("readwrite.txt");
call g2-set-file-position (file, 1, false);
byte-read-1 = call g2-read-byte (file);
```

g2-stream::g2-read-bytes-as-sequence

Reads a specified number of bytes from a stream and returns it as a sequence.

Synopsis

g2-stream::g2-read-bytes-as-sequence
(*stream*: class g2-stream, *n*: integer)
-> *bytes*: sequence

Argument	Description
stream	The G2 stream through which a byte is read.
n	The number of characters to read.

Return Value	Description
bytes	A sequence of bytes.

Description

G2-read-bytes-as-sequence reads a specified number of bytes from a G2 stream and returns it as a sequence. If it is unable to read the stream, it returns an empty sequence.

This version is more useful for binary data.

G2-stream-status

successfully-read-bytes
end-of-file-reached
text-truncated-during-read
successfully-read-bytes
tried-read-bytes-when-closed
error-during-read-bytes
tried-read-bytes-on-stream-not-opened-for-read

g2-stream::g2-read-bytes-as-text

Reads bytes from a stream and returns it as a text.

Synopsis

g2-stream::g2-read-bytes-as-text
(*stream*: class g2-stream, *n*: integer)
-> *bytes*: text

Argument	Description
stream	The G2 stream through which the bytes are read.
n	The number of characters to read.

Return Value	Description
bytes	A text string of the characters read.

Description

G2-read-bytes-as-text reads a specified number of bytes from a G2 stream and returns it either as a text or as -1 if G2 is unable to read it.

G2-stream-status

successfully-read-bytes
end-of-file-reached
text-truncated-during-read
successfully-read-bytes
tried-read-bytes-when-closed
error-during-read-bytes
tried-read-bytes-on-stream-not-opened-for-read

g2-stream::g2-read-line

Reads a line from a file.

Synopsis

g2-stream::g2-read-line
(*stream*: class g2-stream)
-> *line-from-file*: text

Argument	Description
stream	The G2 stream from which a line is read.

Return Value	Description
line-from-file	A text string of a line read from *stream*.

Description

G2-read-line reads a line from a file. You can do random-access read operations by using g2-read-line with the g2-set-file-position procedure if your operating system supports this activity.

G2-stream-status

successfully-read-line
error-during-read-line
end-of-file-reached

Example

This example reads a line from the file `readwrite.txt` and returns the line to line-read-1:

```
file: class g2-stream;
line-read-1:text;
...
file = call g2-open-file-for-read ("readwrite.txt");
call g2-set-file-position (file, 1, false);
line-read-1 = call g2-read-line (file);
```

g2-stream::g2-write-byte

Writes a byte to the file associated with a specified G2 stream.

Synopsis

g2-stream::g2-write-byte
(*stream*: class g2-stream, *new-byte*: integer)

Argument	Description
stream	The G2 stream associated with the file.
new-byte	The byte that is written to the file. The value of *new-byte* must be within the range 0 to 255 inclusive.

Description

G2-write-byte writes a byte to the file associated with the G2 stream.

G2-stream-status

successfully-wrote-byte
error-during-write-byte

Example

The following example writes a byte to the file `readwrite.txt`:

```
file: class g2-stream;
...
file = call g2-open-file-for-read-and-write("readwrite.txt");
call g2-write-byte(file, 99);
```

g2-stream::g2-write-bytes

Synopsis

g2-stream::g2-write-bytes
(*stream*: class g2-stream, *data*: sequence)

Argument	Description
stream	The G2 stream associated with the file.
data	The bytes that are written to the file. The value of *data* is a sequence of 8-bit bytes to be written as binary data and a byte is a value from 0 - 255.

Description

G2-write-bytes writes data to the file associated with the G2 stream.

G2-stream-status

successfully-wrote-byte
error-during-write-byte

g2-stream::g2-write-line

Writes a line to the file associated with a G2 stream.

Synopsis

g2-write-line
(*stream*: class g2-stream, *new-line*: text)

Argument	Description
stream	The G2 stream associated with the file.
new-line	The line to write to the file open on *stream*.

Description

G2-write-line writes *new-line* to the file open on *stream*, then appends the line-delimiting character(s) specified by the external-line-separator attribute of the text-conversion-style associated with *stream*. G2-write-line also translates any linefeed characters in *new-line* into the specified line-delimiting character(s) and writes the translated character(s) to the file in place of the linefeed.

You can add a linefeed character to *new-line* by typing Ctrl + j, by pasting in a linefeed character, or by entering a Return on a scrolling editor.

You can perform random-access writes by using g2-write-line with the g2-set-file-position procedure, if your operating system supports this.

G2-stream-status

successfully-wrote-line
error-during-write-line

Example

This example writes the line "Write this line to the file." to the file `readwrite.txt`:

```
file: class g2-stream;
line: text = "Write this line to the file.";
...
file = call g2-open-file-for-read-and-write("readwrite.txt");
call g2-write-line(file, line);
```

g2-write-line-in-gensym-charset

Writes a line containing Gensym character set characters to the file associated with a G2 stream.

Synopsis

g2-write-line-in-gensym-charset
(*stream*: class g2-stream, *new-line*: text)

Argument	Description
stream	The G2 stream associated with the file.
new-line	The line including Gensym character set characters to write to the file open on *stream*.

Description

G2-write-line-in-gensym-charset writes *new-line* to the file open on *stream*. The bytes written include any Gensym character set characters in *new-line.* For example, if you use g2-write-line-in-gensym-charset to write the text of a procedure to a file, all line breaks visible in the Scrollable Editor appear as `@L` characters in the file. The @L characters represent the two-character sequence for encoding a linefeed character in the Gensym character set.

After writing the line, g2-write-line-in-gensym-charset appends the line-delimiting character(s) specified by the external-line-separator attribute of the text-conversion-style associated with *stream*. It also translates any linefeed characters in *new-line* into the specified line-delimiting character(s) and writes the translated character(s) to the file in place of the linefeed.

You can add a linefeed character to *new-line* by typing Ctrl + j, by pasting in a linefeed character, or by entering a Return on a scrolling editor.

You can perform random-access writes by using g2-write-line with the g2-set-file-position procedure, if your operating system supports this.

G2-stream-status

successfully-wrote-line
error-during-write-line

Example

The following example writes the value of the company-name attribute, which includes a Gensym character trademark symbol, to a file. The file contents will contain the symbol encoded as a Gensym character.

The procedure passes a company name to g2-write-line-in-gensym-charset and writes it to a text file.

```
write-gensym-charset-line(company: class company)
stream: class g2-stream;
send-text: text;
begin
    send-text = the company-name of company;
    stream = call g2-open-file-for-write ("/home/user/company.txt");
    transfer stream to this workspace at (50, 50);
    call g2-write-line-in-gensym-charset(stream, send-text)
end
```

For a description of using the Gensym character set, see G2 Character Support in the *G2 Reference Manual*.

g2-stream::g2-write-string

Writes a string to the file associated with a G2 stream.

Synopsis

g2-stream::g2-write-string
(*stream*: class g2-stream, *new-string*: text)

Argument	Description
stream	The G2 stream associated with the file.
new-string	The string to write to the file open on *stream*.

Description

G2-write-string writes *new-string* to the file open on *stream*. It translates any linefeed characters in *new-string* into the line-delimiting character(s) specified by the external-line-separator attribute of the text-conversion-style associated with *stream*, and writes the translated character(s) to the file in place of the linefeed.

You can add a linefeed character to *new-string* by typing Ctrl + j, by pasting in a linefeed character, or by entering a Return on a scrolling editor.

You can perform random-access writes by using g2-write-string with the g2-set-file-position procedure, if your operating system supports this.

G2-stream-status

successfully-wrote-string
error-during-write-string

Example

This example writes the string "Write this string to the file." to the file readwrite.txt:

```
file: class g2-stream;
string: text = "Write this string to the file.";
...
file = call g2-open-file-for-read-and-write ("readwrite.txt");
call g2-write-string (file, string);
```

g2-write-string-in-gensym-charset

Writes a string containing Gensym character set characters to the file associated with a G2 stream.

Synopsis

g2-write-string-in-gensym-charset
(*stream*: class g2-stream, *new-string*: text)

Argument	Description
stream	The G2 stream associated with the file.
new-string	The string including Gensym character set characters to write to the file open on *stream*.

Description

G2-write-string-in-gensym-charset writes *new-string* to the file open on *stream*. The bytes written include any Gensym character set characters in *new-string*. For example, if you use g2-write-string-in-gensym-charset to write the text of a procedure to a file all line breaks visible in the Scrollable Editor appear as `@L` characters in the file. The @L characters represent the two-character sequence for encoding a linefeed character in the Gensym character set.

G2-write-string-in-gensym-charset translates any linefeed characters in *new-string* into the line-delimiting character(s) specified by the external-line-separator attribute of the text-conversion-style associated with *stream*, and writes the translated character(s) to the file in place of the linefeed.

You can add a linefeed character to *new-string* by typing control-j, by pasting in a linefeed character, or by entering a Return on a scrolling editor.

You can perform random-access writes by using g2-write-string-in-gensym-charset with the g2-set-file-position procedure, if your operating system supports this.

G2-stream-status

successfully-wrote-string
error-during-write-string

9

Get Hierarchy Operations

Describes procedures that obtain KB hierarchies.

Introduction

Use the get hierarchy operations system procedures to provide programmatic access to the Inspect commands that return KB hierarchies.

g2-get-class-hierarchy

Returns the class hierarchy of a class.

Synopsis

g2-get-get-class-hierarchy
(*class-name*: symbol)
-> *class-hierarchy*: structure

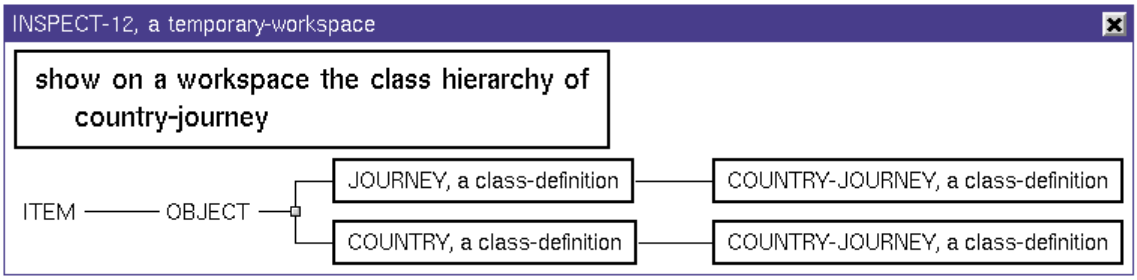
Argument	Description
class-name	A symbol that names a class.

Return Value	Description
class-hierarchy	A structure containing the class hierarchy.

Example

The class used in this example is country-journey. It inherits directly from country and journey classes. Both country and journey inherit directly from object.

Here is the hierarchy as it is shown on an Inspect workspace:



This is the structure returned from the system procedure:

```
structure (item-or-name: "ITEM",
     children: sequence (structure (item-or-name: "OBJECT",
          children: sequence (structure (item-or-name: country,
               children: sequence (structure (item-or-name: country-journey))),
                    structure (item-or-name: journey,
                    children: sequence (structure (item-or-name:
                                                   country-journey)))))))
```

g2-get-containment-hierarchy

Returns a sequence containing the KB items superior to the item argument.

Synopsis

g2-get-containment-hierarchy
(*item*: class item)
-> *containment-hierarchy*: sequence

Argument	Description
item	Specifies the item for which you wish to obtain the superior items in the KB.

Return Value	Description
containment-hierarchy	A sequence of symbols naming items that are superior to *item* in the KB hierarchy. The items are sequenced in bottom-to-top hierarchy order.

Example

In the procedure-calling statement below, china-trip is an instance of a user-defined class which resides on a subworkspace of its class definition, country-journey. The class definition resides on top-level-workspace.

```
containment-hierarchy = g2-get-containment-hierarchy(china-trip)
```

The procedure call returns this sequence:

```
sequence (instance-subworkspace,country-journey, top-level-workspace)
```

g2-get-direct-subclasses

Returns a sequence of symbols naming the direct subclasses of the class-name argument.

Synopsis

g2-get-direct-subclasses
(*class-name*: symbol)
-> *direct-subclasses*: sequence

Argument	Description
class-name	A symbol naming a defined class.

Return Value	Description
direct-subclasses	A sequence of symbols naming the direct subclasses of *class-name*.

g2-get-explanation-hierarchy

Returns a structure describing the hierarchy of rule invocations for a variable or parameter.

Synopsis

g2-get-explanation-hierarchy
(*var-or-param*: variable-or-parameter)
-> *explanation-tree*: structure

Argument	Description
var-or-param	A variable or parameter instance.

Return Value	Description
explanation-tree	A structure describing the hierarchy of rule invocations for *class-name*.

Description

The Explanation Facility allows you to trace rule invocations, as well as variable and parameter updates. For general information about the Explanation Facilities, see Explanation Facilities in the *G2 Reference Manual*.

The syntax of the *explanation-tree* structure is:

structure
(node-type: *symbol*,
item-or-value: *item-or-value*,
node-specific-data: *sequence*,
children: *sequence*)

where:

- node-type is one of the symbols: item, rule, variable-or-paramater, specific-formula, or data-server-or-initial-value.
- item-or-value depends on the node-type, as follows:
  - When node-type is item, rule, or variable-or-parameter, the item.
  - When node-type is specific-formula, the text of the specific formula.
  - When node-type is data-server-or-initial-value, the text "External Data Server" or "Initial Value", as apropriate.

- node-specific-data depends on the node-type, as follows:
  - When node-type is variable-or-parameter, a value-structure as described in variable-or-parameter in Class Dictionary in the *G2 Class Reference Manual*.
  - When node-type is rule, a sequence of structures that describe the bindings of the local variables in the rule, where each structure has this syntax:

    structure
    (local-name: *text*,
    item-or-value: *item-or-value*)

  - For all other values of node-type, node-specific-data is an empty sequence.
- children is a sequence of structures that describe the source of the data for the explanation node, where each structure has the same synax as the *explanation-tree* return value.

  If a circularity is detected, for example, a rule both gets triggered by a variable and also concludes a value to that variable, the variable-or-parameter structure appears as one of the rule's children, with no children of its own, to prevent an infinitely deep structure.

g2-get-method-hierarchy

Returns a structure containing the method hierarchy for a method-declaration.

Synopsis

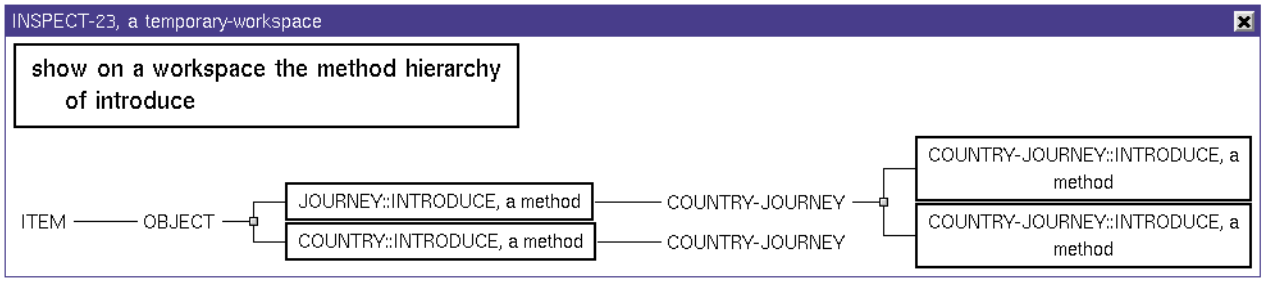
g2-get-method-hierarchy
(*method-declaration-name*: symbol)
-> *method-hierarchy*: structure

Argument	Description
method-declaration-name	A symbol naming a method-declaration.

Return Value	Description
method-hierarchy	A structure containing the method hierarchy.

Example

Here a method hierarchy for a method-declaration named introduce, as shown on an Inspect workspace:



Here is the structure returned by the system procedure for introduce:

```
structure (item-or-name: "ITEM",
    children: sequence (structure (item-or-name: "OBJECT", children:
        sequence (structure (item-or-name: country::introduce,
            children: sequence (structure (item-or-name:
                "COUNTRY-JOURNEY"))),
                structure (item-or-name: journey::introduce,
                    children: sequence (structure (item-or-name:
                        "COUNTRY-JOURNEY",

                    children: sequence (structure (item-or-name: the method
                        having uuid "2512082419ec11d49aea00609703e694"),
                    structure (item-or-name: the method having uuid
                        "2514082419ec11d49aea00609703e694"))))))))
```

g2-get-module-hierarchy

Returns a structure containing the module hierarchy for a module.

Synopsis

g2-get-module-hierarchy
(*module-name*: symbol)
-> *module-hierarchy*: structure

Argument	Description
module-name	The name of a module in the current KB or none to get the entire module hierarchy.

Return Value	Description
module-hierarchy	A structure containing the module hierarchy. This structure is similar to the structure returned by g2-get-class-hierarchy.

g2-get-procedure-caller-hierarchy

Returns a structure containing the caller hierarchy of a procedure. The structure contains the procedure argument and the procedures that call it directly or indirectly.

Synopsis

g2-get-procedure-caller-hierarchy
(*procedure-name*: symbol)
-> *caller-hierarchy*: structure

Argument	Description
procedure-name	A symbol that names a procedure.

Return Value	Description
caller-hierarchy	A structure containing the caller hierarchy.

Example

For this example, procedure3 is called directly by procedure2, and procedure1 calls procedure2.

```
caller-hierarchy =
    call g2-get-procedure-caller-hierarchy(the symbol procedure3)
```

The value of caller-hierarchy is:

```
structure (item-or-name: procedure3,
    children: sequence (structure (item-or-name: procedure2,
        children: sequence (structure (item-or-name: procedure1)))))
```

g2-get-procedure-calling-hierarchy

Returns a structure containing the calling hierarchy. The structure contains the procedure argument and the procedures it calls directly or indirectly.

Synopsis

g2-get-procedure-calling-hierarchy
(*procedure-name*: symbol)
-> *calling-hierarchy*: structure

Argument	Description
procedure-name	A symbol that names a procedure.

Return Value	Description
calling-hierarchy	A structure containing the calling hierarchy.

Example

In this example, procedure1 calls procedure2, and procedure2 calls procedure3:

```
calling-hierarchy =
    call g2-get-procedure-calling-hierarchy(the symbol procedure1)
```

The value of calling-hierarchy is:

```
structure (item-or-name: procedure1,
    children: sequence (structure (item-or-name: procedure2,
        children: sequence (structure (item-or-name: procedure3)))))
```

g2-get-procedure-invocation-hierarchy

Returns a structure containing the current procedure invocations on the calling stack.

Synopsis

g2-get-procedure-invocation-hierarchy
( )
-> *invocation-hierarchy*: structure

Return Value	Description
invocation-hierarchy	A structure containing the procedure invocation hierarchy.

Example

In this example, proc1 is the on the top of the calling hierarchy. The procedure hierarchy-test receives this structure from g2-get-procedure-invocation-hierarchy:

```
structure (item-or-name: "top-level",
     children: sequence (structure (item-or-name: "proc1( )
          no local names available",
               children: sequence (structure (item-or-name: "proc2( )
                    no local names available",
                         children: sequence (structure (item-or-name: "proc3( )
                              st: structure = no value",
                                   children: sequence (structure (item-or-name:
                                        "hierarchy-test( )
                                        struct: structure = no value;
                                        seq: sequence = no value",
                                             children: sequence (structure (item-or-name:
                                                  "g2-get-procedure-invocation-
                                                       hierarchy( )
                                                       hierarchy:
                                                            structure = no value")))))))))))
```

g2-get-strict-instances-of-class

Returns a sequence of symbols naming the items that are strict instances of the class argument (excludes instances of subclasses of the class argument).

Synopsis

g2-get-strict-instances-of-class
(*class-name*: symbol)
-> *strict-instance*: sequence

Argument	Description
class-name	A symbol that names a defined class.

Return Value	Description
strict-instances	A sequence of symbols naming the items that are strict instances of the class.

g2-get-top-level-workspaces

Returns a sequence of symbols naming the top-level workspaces in the KB.

Synopsis

g2-get-top-level-workspaces
( )
-> *workspaces*: sequence

Return Value	Description
workspaces	A sequence of symbols naming the top-level workspaces in the KB.

g2-get-workspace-hierarchy

Returns a structure containing the workspace hierarchy of a workspace.

Synopsis

g2-get-workspace-hierarchy
(*ws-name*: symbol)
-> *workspace-hierarchy*: structure

Argument	Description
ws-name	A symbol that names a kb-workspace or none to get the entire workspace hierarchy.

Return Value	Description
workspace-hierarchy	A structure containing the workspace hierarchy.

Example

In this example, the procedure is called with a top-level workspace that has two direct subworkspaces. This structure is returned:

```
structure (item-or-name: top-level-ws,
    children: sequence (structure (item-or-name: first-subworkspace),
                        structure (item-or-name: second-subworkspace)))
```

10

G2 Graphical Language (G2GL)

Describes the system procedure for using the G2 Graphical Language (G2GL).

Introduction

Use the G2 Graphical Language (G2GL) system procedures to interact with G2GL processes and process instances. For more information, see Part V, G2 Graphical Language (G2GL) in the *G2 Reference Manual*.

g2-call-g2gl-process-as-procedure

Calls a G2GL process as a procedure, with arguments and return values.

Synopsis

g2-call-g2gl-process-as-procedure
(*process*: class g2gl-process, *argument-list*: sequence
-> *return-values*: sequence)

Argument	Description
process	The g2gl-process to call as a procedure.
argument-list	A sequence of G2GL argument variables for the process.

Return Value	Description
return-values	A sequence of the return values to the process.

Description

If you define a G2GL process with arguments and/or return values, you must call it as if it were a procedure, which allows you to pass arguments and return values.

This procedure compiles the G2GL process first if it has not already been compiled, then calls it.

This procedure returns a single value, the sequence of values returned by the g2gl-process, or signals an error if a fault is not handled at the top-level of the g2gl-process. The error is an instance of the new system error class g2gl-fault, which is a subclass of g2-error. The error-description has a text describing the fault, which is the same as the message shown on the breakpoint execution display. The error item has two additional attributes, fault-name, a symbol, and fault-data, which is "none" for all non-user faults and for user faults that don't include fault data.

If the G2GL process does not have arguments or return values, use g2-execute-g2gl-process.

g2-collect-all-g2gl-process-instances

Returns a list of all G2GL process instances associated with a process.

Synopsis

g2-collect-all-g2gl-process-instances
(*process*: g2gl-process)
-> *process-instances*: sequence

Argument	Description
process	The g2gl-process whose process instances to collect.

Return Value	Description
process-instaces	A sequence of g2gl-process-instance instances for the g2gl-process.

g2-compile-g2gl-process

Compiles a G2GL process.

Synopsis

g2-compile-g2gl-process
(*process*: class g2gl-process)
-> *success*: truth-value

Argument	Description
process	The g2gl-process to compile.

Return Value	Description
success	Returns true if the process compiles successfully, false otherwise.

Description

Once you have defined a G2GL process, you must compile it before it can be instantiated. The latest-attempted-compilation-version-number attribute of the G2GL process increments each time it is compiled.

If you change the process, you must recompile it in order to use the latest version for the execution. If you change the process and do not recompile it, the process uses the existing compilation version for its execution.

When you compile, if an old compilation version is still being used to execute a process instance, the old version continues to exist until all process instances that are based on that version terminate. This means that, in theory, there could be any number of distinct versions of the process executing concurrently. However, all new instances of the process use the latest version that has been successfully compiled.

G2GL detects compilation errors and warnings, and displays them in the process body. When the same error or warning occurs more than once in a G2GL process, for example, when referring to an undeclared variable, G2GL creates compilation errors and warnings for each occurrence of the error or warning.

The process also keeps track of the number of errors and warnings in the number-of-errors-in-latest-attempted-compilation and number-of-warnings-in-latest-attempted-compilation attributes.

g2-execute-g2gl-process

Executes a G2GL process programmatically from a G2 procedure. The G2GL process cannot have arguments or return values.

Synopsis

g2-execute-g2gl-process
(*process*: class g2gl-process)
-> *process-instance*: g2gl-process-instance

Argument	Description
process	The g2gl-process to execute.

Return Value	Description
process-instance	An instance of g2gl-process-instance that represents the running process.

Description

This procedure compiles the G2GL process first if it has not already been compiled, then executes it.

If the G2GL process has arguments or return values, use g2-call-g2gl-process-as-procedure.

g2-export-g2gl-process-as-xml

Exports a G2GL process to an XML document.

Synopsis

g2-export-g2gl-process-as-xml
(*process*: class g2gl-process, *file-path*: text)

Argument	Description
process	The g2gl-process to execute.
file-path	A complete path name of the XML file to export.

Description

The structure of the XML document corresponds to the BPEL specification.

G2GL extensions to BPEL are exported to the `http://gensym.com/g2gl/` XML namespace URI and use the `g2gl:` prefix. G2GL uses the `http://gensym.com/g2gl/g2gl-expression` namespace for the `expressionLanguage` and `queryLanguage` attributes.

When exporting a g2gl-process as XML, the value of the g2gl-namespace-map attribute is converted to a set of XML namespace declarations on the process element.

The g2gl-target-namespace attribute on g2gl-process corresponds to the `targetNamespace` attribute on the `<process>` element. It is initialized to an empty string when you create a g2gl-process in G2.

g2-export-g2gl-process-as-xml-text

Exports a G2GL process to an XML document as text.

Synopsis

g2-export-g2gl-process-as-xml-text
(*process*: class g2gl-process)
-> *document*: text

Argument	Description
process	The g2gl-process to execute.

Return Value	Description
document	A G2 text that is the XML text of the g2gl-process.

Description

The structure of the XML text corresponds to the BPEL specification.

The returned G2 text contains the entire XML document. Exporting to a text is significantly faster than exporting to a file. The maximum size of the text containing the XML code is 1 MB. The procedure must complete before the scheduler allows other processing to occur.

G2GL extensions to BPEL are exported to the `http://gensym.com/g2gl/` XML namespace URI and use the `g2gl:` prefix. G2GL uses the `http://gensym.com/g2gl/g2gl-expression` namespace for the `expressionLanguage` and `queryLanguage` attributes.

When exporting a g2gl-process as XML, the value of the g2gl-namespace-map attribute is converted to a set of XML namespace declarations on the process element.

The g2gl-target-namespace attribute on g2gl-process corresponds to the `targetNamespace` attribute on the `<process>` element. It is initialized to an empty string when you create a g2gl-process in G2.

g2-import-g2gl-process-from-xml

Creates a G2GL process by importing from an XML document.

Synopsis

g2-import-g2gl-process-from-xml
(*file-path*: text)
-> *process*: class g2gl-process

Argument	Description
file-path	A complete path name to the XML file to import.

Return Value	Description
process	A g2gl-process instance that corresponds with the XML file.

Description

The structure of the XML document corresponds to the BPEL specification.

For details, see the *G2 Reference Manual*.

Note Currently, G2GL only supports the G2GL expression language. It does not support the standard BPEL expression language, XPath, or any other expression language. When importing from XML documents, G2GL discards expressions in non-G2GL languages.

g2-import-g2gl-process-from-xml-text

Creates a G2GL process by importing from an XML text.

Synopsis

g2-import-g2gl-process-from-xml-text
(*document*: text)
-> *process*: class g2gl-process

Argument	Description
document	A G2 text that is the XML code to import.

Return Value	Description
process	A g2gl-process instance that corresponds with the XML text.

Description

The structure of the XML text corresponds to the BPEL specification.

For details, see the *G2 Reference Manual*.

The G2 text that contains the XML text contains the entire XML document. Importing from a text is significantly faster than importing from a file. The maximum size of the text containing the XML code is 1 MB. The procedure must complete before the scheduler allows other processing to occur.

g2-kill-all-g2gl-process-instances

Deletes all executing G2GL process instances for a G2GL process.

Synopsis

g2-kill-all-g2gl-process-instances
(*process*: class g2gl-process)

Argument	Description
process	The g2gl-process whose process instances to delete.

g2-kill-g2gl-process-instance

Deletes an individual G2GL process instance.

Synopsis

g2-kill-g2gl-process-instance
(*process-instance*: class g2gl-process-instance)

Argument	Description
process-instance	The g2gl-process-instance to delete.

g2-pause-g2gl-process-instance

Pauses a G2GL process instance.

Synopsis

g2-pause-g2gl-process-instance
(*process-instance*: class g2gl-process-instance)

Argument	Description
process-instance	The g2gl-process-instance to pause.

Description

Pausing a G2GL process instances sets the g2gl-process-instance-is-paused attribute of the process instance to true.

g2-resume-g2gl-process-instance

Resumes a G2GL process instance that is currently paused.

Synopsis

g2-resume-g2gl-process-instance
(*process-instance*: class g2gl-process-instance)

Argument	Description
process-instance	The g2gl-process-instance to resume.

Description

Resuming a G2GL process instances sets the g2gl-process-instance-is-paused attribute of the process instance to false.

11

Hash Table Operations

Describes the system procedure for using G2 hash tables.

Introduction

The hash-table class provides a data structure for fast lookup of a value, based on a key. The key and value can be any item-or-value. When specifying keys as text, the text is case sensitive.

Hash tables provide fast lookups for various types of data, regardless of the number of entries in the table, where the lookup time is proportional to the log of the number of key-value pairs in the table. Use the hash table system procedures for adding and removing elements, accessing values, and dynamically changing values in a hash table.

G2 provides procedures for getting and setting values given a key, clearing individual values from the table given a key, clearing all key-value pairs from the table, and converting hash tables to sequences to allow iterating over the elements.

While sequences and structures also provide the ability to define key-value pairs, structures require that the keys be symbols. Hash tables do not have this restriction; the key and the value can be any G2 item or value. Also, when the number of key-value pairs in a sequence or structure becomes very large, finding elements can be very slow. Thus, when all you require is a set of key-value pairs, we recommend that you use hash tables rather than sequences or structures. If you require the ability to parse the key-value pairs sequentially, you should use a sequence or a list.

G2 provides hash tables as objects, which you can create from the KB Workspace > New Object menu or programmatically, using the create action.

Note G2 does not save the contents of a hash table in a KB when it is saved. When a new KB is loaded, all hash tables are emptied. Also, disabling a hash table empties its contents.

---

g2-set-hash-table-value

Associates a value with a key in a hash table.

Synopsis

g2-set-hash-table-value
(*table*: class hash-table, *key*: item-or-value, *value*: item-or-value)

Argument	Description
table	The hash-table whose key-value pair to set.
key	An item-or-value to set as the key.
value	An item-or-value to set as the value.

Description

The key and value can be any G2 item-or-value.

g2-get-hash-table-value

Associates a value with a key in a hash table.

Synopsis

g2-get-hash-table-value
(*table*: class hash-table, *key*: item-or-value)
-> (*result*: item-or-value, *found*: boolean)

Argument	Description
table	The hash-table that defines the specified *key*.
key	The key whose value to get.

Return Value	Description
result	The value associated with the specified key or false if the key does not exist.
found	True if the specified key exists, false otherwise.

Description

If the specified key is associated with a value in the table, this procedure returns the value and true, indicating that the value was found. If the specified key does not have an associated value, this procedure returns false and false as the two return values.

The specified *key* is considered to be the same as a key in the table if:

- The keys are items and the items are identical.
- The keys are of type text and the texts have the same length and contain the same characters. Note that comparison of texts is case sensitive.
- The keys are any value other than a text and the values are equal.

g2-clear-hash-table-value

Removes a value associated with a key in a hash table.

Synopsis

g2-clear-hash-table-value
(*table*: class hash-table, *key*: item-or-value)

Argument	Description
table	The hash-table that defines the specified *key*.
key	The key whose value to clear.

Description

This procedure removes the association of the specified *key* with a value in the table. If the key is not currently associated with a value, this procedure does nothing.

g2-clear-hash-table

Removes all key-value pairs from the table.

Synopsis

g2-clear-hash-table
(*table*: class hash-table)

Argument	Description
table	The hash-table whose key-value pairs to remove.

Description

Removes all key-value pairs from the table.

g2-hash-table-to-sequence

Returns a sequence of alternating key-value pairs from a hash table.

Synopsis

g2-hash-table-to-sequence
(*table*: class hash-table)
-> *sequence*: sequence

Argument	Description
table	The hash-table whose key-value pairs to get.

Description

Note The g2-hash-table-to-sequence procedure is not supported and will be removed in a future release. The procedure exists for compatibility with G2 Version 8.0 Beta Rev. 0 only. Use the hash-table-sequence hidden attribute on hash-table instances instead.

12

History Clearing Operation

Describes the system procedure that clears history values.

Introduction

Use the history clearing system procedure for clearing history values in variables and parameters.

g2-clear-histories

Clears all history values from a variable or a parameter.

Synopsis

g2-clear-histories
(*variable-or-param-with-history*: class variable-or-parameter)

Argument	Description
variable-or-param-with-history	The variable or parameter whose history to clear.

Description

Calling this procedure clears all history values from the variable or parameter you pass as the single argument.

g2-initialize-parameter

Clears all history values from a parameter and sets the initial value and collection time.

Synopsis

g2-initialize-parameter
(*parameter*: class parameter, *value*: value, *collect-time*: quantity)

Argument	Description
parameter	The parameter whose history to initialize.

Description

Initializes *parameter* to have only one historical value. The value, initial-value, and last-recorded-value are all set to the specified *value* and *collection-time*.

13

Indexed Attribute Operations

Describes the procedure that retrieves the class instances with a particular value for an indexed attribute.

Introduction

Use the indexed attribute system procedure to return a list of every item with the **indexed attribute** value that you specify.

g2-indexed-attribute-item-list

Determines all instances of a class with the specified attribute-value.

Synopsis

g2-indexed-attribute-item-list
(*indexed-item-list*: class item-list, *class-name*: symbol,
attribute-name: symbol, *attribute-value*: value)

Argument	Description
indexed-item-list	An item list that receives the items whose *attribute-name* attribute has the value specified by *attribute-value*. This item list must exist at the time when g2-indexed-attribute-item-list is called; the procedure does not create the item list. For information about how to create lists, see Lists and Arrays in the *G2 Reference Manual*.
class-name	The class of the objects that are searched for the specified *attribute-value*.
attribute-name	The name of the attribute.
attribute-value	The value of *attribute-name* to search for. This procedure returns all items whose *attribute-name* attribute has this value.

Description

G2-indexed-attribute-item-list enables you to find all instances of a class that have the specified *attribute-value*. The procedure inserts any instances with this attribute value at the end of *indexed-item-list*.

The attribute specified by *attribute-name* must be an indexed attribute. Only user-defined classes can have attributes that are indexed. You specify the attribute as:

```
with an index
```

For example, the attribute temp is an indexed attribute:

```
temp initially is 98, with an index
```

Only the following types of attributes can be indexed:

- Attributes defined as the type integer, text, symbol, or truth-value.
- Attributes given by an integer-parameter, text-parameter, symbolic-parameter, or logical-parameter.

Note Indexed attributes require a significant amount of memory, because G2 maintains a hash table for the indexed attributes within a KB. Thus, it is recommended practice to use indexed attributes only when your G2 application requires the rapid access to data provided by hash tables.

---

g2-get-item-of-class-by-attribute-value

Find the first instances of a class with the specified attribute-value.

Synopsis

g2-get-item-of-class-by-attribute-value
(*class-name*: symbol,
attribute-name: symbol, *attribute-value*: value)
-> *item-or-not-found*: item-or-value

Argument	Description
class-name	The class of the objects that are searched for the specified *attribute-value*.
attribute-name	The name of the attribute.
attribute-value	The value of *attribute-name* to search for. This procedure returns all items whose *attribute-name* attribute has this value.

Return Value	Description
item-or-not-found	An item-or-value that identifies the result. If the procedure doesn't find the item of specific class and attribute value, then the symbol item-not-found is returned. If there're multiple items matching the criteria, then the first item found is returned.

Description

G2-get-item-of-class-by-attribute-value enables you to find the first instance of a class that have the specified *attribute-value*. The procedure is almost the same with another related procedure g2-indexed-attribute-item-list but only one item will be returned if there're multiple items matching the criteria.

The attribute specified by *attribute-name* must be an indexed attribute. Only user-defined classes can have attributes that are indexed. You specify the attribute as:

```
with an index
```

For example, the attribute temp is an indexed attribute:

```
temp initially is 98, with an index
```

Only the following types of attributes can be indexed:

- Attributes defined as the type integer, text, symbol, or truth-value.
- Attributes given by an integer-parameter, text-parameter, symbolic-parameter, or logical-parameter.

Note Indexed attributes require a significant amount of memory, because G2 maintains a hash table for the indexed attributes within a KB. Thus, it is recommended practice to use indexed attributes only when your G2 application requires the rapid access to data provided by hash tables.

This procedure is only available in G2 2011 Enterprise Edition.

14

Inspect Operations

Describes procedures that give you programmatic access to Inspect commands.

Introduction

Use the inspect system procedures to programmatically execute the replace, write to the file, recompile, and show on a workspace Inspect commands.

G2 also supplies procedures for obtaining KB hierarchies as described in Get Hierarchy Operations.

g2-abort-inspect-session

Aborts the Inspect session started by a call to g2-start-inspect-session.

Synopsis

g2-abort-inspect-session
(*session-id*: integer)

Argument	Description
session-id	An integer value returned by g2-start-inspect-session which uniquely identifies the session.

Description

This procedure aborts the Inspect session identified by *session-id*.

See g2-start-inspect-session for how to begin an Inspect session.

g2-pause-inspect-session

Pauses the Inspect session started by a call to g2-start-inspect-session.

Synopsis

g2-pause-inspect-session
(*session-id*: integer)

Argument	Description
session-id	An integer value returned by g2-start-inspect-session which uniquely identifies the session.

Description

This procedure pauses the Inspect session identified by *session-id*.

See g2-resume-inspect-session for how to resume an inspect session, and g2-start-inspect-session for how to begin an inspect session.

g2-resume-inspect-session

Resumes the Inspect session started by a call to g2-start-inspect-session and paused by g2-pause-inspect-session.

Synopsis

g2-resume-inspect-session
(*session-id*: integer)

Argument	Description
session-id	An integer value returned by g2-start-inspect-session which uniquely identifies the session.

Description

This procedure resumes the previously paused Inspect session identified by *session-id*.

See g2-pause-inspect-session for how to pause an inspect session, and g2-start-inspect-session for how to begin an inspect session.

g2-run-inspect-command

Executes Inspect commands that replace text, recompile items, and write information to a file.

Synopsis

g2-run-inspect-command
(*command*: text)

Description

This procedure gives you programmatic access to those Inspect commands that produce changes to your kb or write item information to a file. It does not support commands that display items or tables on a temporary workspace or go to an item location.

You can call this procedure to accomplish the actions of these commands only:

- replace
- write to the file
- recompile

The procedure has no return value, and G2 gives no notification that command execution has made changes to your KB or written information to a file.

Note There is a difference in file contents between using this procedure or an interactive Inspect session to write information to a file. The file generated by an interactive Inspect session includes this text: "**Command:*command*"; the file generated by this procedure omits the command text.

See The Inspect Facility in the *G2 Reference Manual* for the complete syntax of the replace, write to a file, and recompile commands. See g2-get-workspace-hierarchy for information on how to execute show on a workspace commands that return matching items in a sequence data structure.

Examples

Here are some example calls:

```
call g2-run-inspect-command("replace phone with fax in every free-text")

call g2-run-inspect-command("write to the file @"/home/user/query-output@"
     the address of customer101")

call g2-run-inspect-command("recompile every procedure")
```

g2-start-inspect-session

Executes the show on a workspace Inspect commands which collect items matching the command description.

Synopsis

g2-start-inspect-session
(*command-string*: text, *callback:* class procedure, *user-data:* item-or-value,
priority: integer, *task-interval:* float)
-> *session-id*: value

Argument	Description
command-string	A quoted show on a workspace Inspect command. Its syntax is: show on a workspace { *item* | every *item filter* }
callback	A user-defined procedure. The g2-start-inspect-session system procedure returns a sequence of the matching items to this procedure. See Description below for the required arguments.
user-data	A item-or-value you pass to g2-start-inspect-session. G2-start-inspect-session passes *user-data* back to your callback procedure.
priority	An integer from 1 - 10.
task-interval	A non-negative float value that specifies the task scheduling interval for the inspect session. G2's default task interval is 1.0.

Return Value	Description
session-id	An integer value that identifies the inspect session.

Description

This procedure executes the show on a workspace Inspect commands which collect items that match the command description. It returns the matching items

to a user-defined callback procedure in a sequence value. The empty sequence is returned when no item matched the description.

For each call to g2-start-inspect-session, there are one or more calls to your callback procedure. The number of callbacks depends on how many items G2 must examine, the complexity of the *filter*, the values of the *priority* and *task-interval* arguments, and what other tasks G2 must perform.

Your callback procedure must accept these arguments:

Argument	Description
inspect-state	A symbol G2 uses to inform you of the progress of the Inspect command. Examples are frames-remaining and finished.
total-items-to-examine	An integer representing the total number of items G2 will examine during the Inspect session.
number-examined-so-far	An integer representing the total number of items G2 has examined so far in the Inspect session.
number-of-items-found	An integer representing the total number of matching items so far in the Inspect session.
matching-items	A sequence containing the matching items found since the last callback procedure call or since the start of g2-start-inspect-session and the first callback procedure call.
user-data	Any item-or-value you wish passed to your callback procedure. For example, you can define *user-data* to be an item-list which your callback procedure code uses to accumulate the matching items returned by all calls to your callback procedure.

These system procedures pause, resume, and abort the inspect session started by a call to g2-start-inspect-session:

- g2-pause-inspect-session.
- g2-resume-inspect-session.
- g2-abort-inspect-session.

See The Inspect Facility in the *G2 Reference Manual* for the complete syntax of the show on a workspace Inspect commands. To programmatically execute the

Inspect commands that replace text, recompile items, and write information to a file, see g2-run-inspect-command.

Examples

Here is an example procedure that calls g2-start-inspect-session. The inspect session identifier is stored in an integer parameter so that it is available for calling g2-pause-inspect-session, g2-resume-inspect-session, and g2-abort-inspect-session.

```
start-inspect-command ( )
id :integer;
begin
    id = call g2-start-inspect-session ("show on a workspace every free-text
                containing the word financial", inspect-callback-procedure,
                inspect-item-list, 3, 1.0);
    conclude that the-session-id-integer-parameter = id
end
```

An example of a callback procedure:

```
inspect-callback-procedure(inspect-state: symbol,
                                    total-items-to-examine: integer,
                                    number-examined-so-far: integer,
                                    number-of-matching-items: integer,
                                    matching-items: sequence,
                                    user-data: item-or-value)
matching-items-length, index: integer;
begin
    matching-items-length = the number of elements in matching-items;
    post "[number-examined-so-far] items have been examined so far out of
            [total-items-to-examine] total items to examine";
    for index = 0 to matching-items-length - 1 do
        insert matching-items[index] at the end of inspect-item-list
    end;
    if inspect-state is finished then
        post "Item collection is complete.
        [number-of-matching-items] matching-items have been found."
end
```

15

KB and Module Operations

Describes procedures for saving, loading, merging, and warmbooting KBs, and for creating, saving, and deleting modules.

Introduction

Use the KB and module system procedures to:

- Load and save KBs
- Create and delete modules
- Save snapshots of the current KB and warmboot from a snapshot file

g2-check-for-consistent-modularization

Checks for consistent modularization of the KB.

Synopsis

g2-check-for-consistent-modularization
( )
-> *return-value*: sequence

Checks the current KB for consistent modularization.

Argument	Description
return-value	A sequence of sequences, where each sequence can contain text strings and items that describe the KB modularization.

Description

The return value is a sequence of sequences, where the text strings are the same as those that G2 returns when using the check for consistent modularization Inspect command. For example, one sequence might contain a text string about classes not being properly modularized followed by a list of offending classes. If the KB is consistently modularized, the procedure returns an empty sequence.

Example

This procedure returns the following sequence when the KB is not consistently modularized:

CHECK-MODULARIZATION

```
check-modularization()
v: sequence;
begin
   v = call g2-check-for-consistent-
     modularization ();
   post "[v]";
end
```

```
#11   2:59:55 p.m.   sequence (sequence
  ("Workspaces belong to modules that are not
  required by the KB; see the installed module-
  information system table for further information.",
    "Modules that exist are not required by the
      KB; see the installed module-information
      system table for further information."))
```

g2-create-module

Creates a module in the current KB.

Synopsis

g2-create-module
(*module-name*: symbol)

Argument	Description
module-name	The name of the module in your current KB that you wish to create programmatically.

Description

This procedure creates a module named *module-name*, subject to the following restrictions:

- A module cannot be created in Runtime systems.
- A module cannot be created in Embedded systems.
- The *module-name* cannot:
  - Duplicate the name of an existing module.
  - Be a reserved word in G2.
  - Be the symbol unspecified.

On successful execution, g2-create-module creates a set of system tables for the new module. The top-level-module attribute of the Module Information system table is *module-name*. All other attributes have default values.

If any of the above restrictions is violated, g2-create-module:

- Leaves the KB unchanged.
- Signals an abort type error.

g2-delete-module

Deletes a module from the current KB.

Synopsis

g2-delete-module
(*module-name*: symbol, *also-delete-associated-workspace*: truth-value)

Argument	Description
module-name	The name of the module in your current KB that you wish to delete programmatically.
also-delete-associated-workspace	Specifies whether to delete any workspaces associated with the deleted module.

Description

This system procedure deletes a module from the current KB programmatically while the KB is running. KB processing halts while the deletion is underway.

When deleting a module, you can optionally delete any workspaces associated with it. For more information about modules, see Modularized KBs in the *G2 Reference Manual*.

After module deletion, any remaining modules that require the deleted module may be inconsistently modularized. Be sure to remove the name of the deleted module from the list of requirements for any other modules.

Example

This example specifies that G2 should delete the workspaces associated with the module. Notice that the symbol precedes the module name.

```
call g2-delete-module(the symbol support-module, true)
```

g2-load-kb

Loads a new KB.

Synopsis

g2-load-kb
(*pathname-string*: text, *resolve-conflicts-automatically*: truth-value, *bring-formats-up-to-date*: truth-value)
-> *kb-loaded*: truth-value

Argument	Description
pathname-string	The name of the KB file to load programmatically.
resolve-conflicts-automatically	Specifies whether to resolve conflicts automatically when G2 loads the KB.
bring-formats-up-to-date	Specifies whether to bring internal formats up to date automatically when G2 loads the KB. For example, if G2 were to change an internal format, such as the width of the indent for wrapped text lines in the Text Editor, you could update the KB to that new format by providing a true value for this argument. A value of false leaves the KB in its format after the last Save KB command.

Return Value	Description
kb-loaded	A truth-value returning true if the KB is loaded successfully or false if it is not.

Description

Use this system procedure to load a KB from within a procedure or rule. KB processing halts while the procedure loads a new KB, clearing the current one. The procedure signals an error and returns false if it is unable to load the new KB.

Note G2 can never return true for a successful load, because loading a new KB clears the existing KB, including the calling procedure.

Example

An example of using the g2-load-kb system procedure is:

```
return-truth-value = call g2-load-kb("/home/ghw/kbs/example.kb", true, true)
```

g2-merge-kb

Merges a KB into the current KB.

Synopsis

g2-merge-kb
(*pathname-string*: text, *resolve-conflicts-automatically*: truth-value, *bring-formats-up-to-date*: truth-value, *install-system-tables*: truth-value)
-> *kb-merged*: truth-value

Argument	Description
pathname-string	The file name of the KB to merge into the current KB programmatically.
resolve-conflicts-automatically	Specifies whether to resolve conflicts automatically when G2 merges the KB.
bring-formats-up-to-date	Specifies whether to bring internal formats up to date automatically when G2 loads the KB. For example, if G2 were to change an internal format, such as the width of the indent for wrapped text lines in the Text Editor, you could update the KB to that new format by providing a true value for this argument. A value of false leaves the KB in its format after the last Save KB command.
install-system-tables	Specifies whether to install the system tables of the merged KB.

Return Value	Description
kb-merged	A truth value that is true if the KB is merged successfully or false if it is not.

Description

This system procedure merges in a KB programmatically, pausing only to merge in the new KB, and resuming processing automatically after the procedure completes, regardless of whether it was successful.

The system procedure returns true upon a successful merge, or false if it is unable to complete the merge. G2 displays messages and signals any errors if necessary.

This procedure performs the same operation as the Merge KB menu choice.

Example

An example of using the g2-merge-kb system procedure, specifying each of the merge option arguments as true, is:

```
return-truth-value =
    call g2-merge-kb("/home/ghw/kbs/example.kb", true, true, true)
```

g2-merge-kb-ex

Merges a KB into the current KB, with additional options.

Synopsis

g2-merge-kb-ex
(*pathname*: text, *options*: structure)
-> *kb-merged*: truth-value

Argument	Description
pathname	The file name of the KB to merge into the current KB programmatically.
options	A structure with the following syntax: structure (resolve-conflicts-automatically: *truth-value*, bring-formats-up-to-date: *truth-value*, merge-kb: *truth-value*, install-system-tables: *truth-value*, file-progress-display: *truth-value*, restore-workspaces: *truth-value*, post-logbook-messages: *truth-value*, show-conflicts: *truth-value*, disable-ui-during-load: *truth-value*) See Description for a description of each option in the structure.

Return Value	Description
kb-merged	A truth value that is true if the KB is merged successfully or false if it is not.

Description

You can use the system procedure to load KBs for end user applications to avoid showing Operator Logbook messages, default workspaces, conflict workspaces, and file progress. It also allows you to disable the user interface while loading.

The *options* structure has these attributes:

- resolve-conflicts-automatically — Resolves conflicts automatically if true. The default is true.
- bring-formats-up-to-date — Brings formats up to date if true. The default is false.
- merge-kb — Merges the KB into the existing KB if true, or clears the current KB before merging if false. The default is true.
- install-system-tables — Installs system tables if true. The default is false.
- file-progress-display — Shows a progress bar while merging the KB if true. The default is true.
- restore-workspaces — Shows the default workspaces from the KB if true. The default is true.
- post-logbook-messages — Posts Operator Logbook messages while loading the KBs if true. The default is true.
- show-conflicts — Posts conflict workspaces and messages to the Operator Logbook while loading the KB if true. The default is true.
- disable-ui-during-load — Disables user interaction and shows hourglass cursor if true. The default is false.

Example

The following code fragment merges the KB named `myapp.kb` without displaying a progress bar, default workspaces, logbook messages, or conflict workspaces, and disables user interaction during loading:

```
start g2-merge-kb-ex ("myapp.kb",
    structure
    (file-progress-display: false,
    restore-workspaces: false,
    post-logbook-messages: false,
    show-conflicts: false,
    disable-ui-during-load: true))
```

g2-save-kb

Saves the current KB.

Synopsis

g2-save-kb
(*pathname*: text, *file-progress-display*: truth-value, *window*: item-or-value)
-> *kb-saved*: truth-value

Argument	Description
pathname	The name of the KB that you wish to save programmatically.
file-progress-display	Specifies whether to show the file-progress display while the KB is being saved.
window	Enter a g2-window object to save the existing workspace layout of a window or false if you do not wish to save the current layout.

Return Value	Description
kb-saved	A truth value that is true if the KB is saved successfully or false if it is not.

Description

Lets you save a KB programmatically. This procedure performs the same operations as the Save KB menu choice.

Note If you use g2-save-kb to save a modularized KB, the system procedure will save the KB as an All file, even if the KB is correctly modularized. To save a modularized KB, use g2-save-module.

Caution

G2 enters a wait state during execution of this procedure, which allows other processing that can asynchronously change the G2 context. Be sure to revalidate the context as needed after the procedure returns.

Saving the Current Workspace Layout

You can save the current workspace layout by specifying a g2-window object as the third argument of this procedure. However, while you can name a g2-window, you cannot cause that name to persist through a KB save and reload operation. Therefore, if you name and use a g2-window name as an argument to this procedure, the name will not exist after reloading the KB. The unique properties of G2 window items are described in G2-Windows in the *G2 Reference Manual*.

g2-save-kb-without-other-processing

Saves the current KB without interruption as an all KB.

Synopsis

g2-save-kb-without-other-processing
(*pathname*: text, *file-progress-display*: truth-value, *win-or-none*: item-or-value)
-> *kb-saved*: truth-value

Argument	Description
pathname	The pathname of the KB that you wish to save programmatically.
file-progress-display	Specifies whether to show the file-progress display while the KB is being saved.
win-or-none	Enter a g2-window object to save the existing workspace layout of a window or none if you do not wish to save the current layout.

Return Value	Description
kb-saved	A truth value that is true if the KB is saved successfully or false if it is not.

Description

Saves a KB by first pausing G2, saving your KB, and then resuming G2. This procedure and g2-save-kb write your KB to a single file even when it is correctly modularized. Use g2-save-module-without-other-processing or g2-save-module to save your KB in separate module files.

Saving the Current Workspace Layout

When you specify a g2-window object, G2 saves the current workspace layout. However, while you can name a g2-window and use that name for the *window-or-not* argument, a g2-window does not persist through KB saving and reloading. A reference to the name of a g2-window that no longer exists elicits frame notes. Consider using an indirect reference to refer to the g2-window associated with your interaction with G2. The unique properties of G2 window items are described in G2-Windows in the *G2 Reference Manual*.

g2-save-module

Saves a module from the current KB, optionally with all required modules.

Synopsis

g2-save-module
(*pathname*: text, *file-progress-display*: truth-value, *win-or-none*: item-or-value, *module*: symbol, *include-required-modules*: truth-value)
-> *module-saved*: truth-value

Argument	Description
pathname	The full path and file name of the file where the module will be saved.
file-progress-display	Specifies whether to show the file-progress display while the module is being saved.
win-or-none	Enter a g2-window object to save the existing workspace layout of a window or none if you do not wish to save the current layout. For additional information about this argument, see Saving the Current Workspace Layout.
module	The name of the module in your current KB that you wish to save programmatically.
include-required-modules	Specifies whether to save any required modules. Specifying this argument as true saves the module with any modules it requires.

Return Value	Description
module-saved	A truth-value that is true if every module is saved successfully, or false if any module is not saved, for example, because its file is read-only.

Description

This system procedure saves a module to the specified path, and optionally saves all required modules as well. When saving required modules, the procedure saves each one in the directory specified in the first argument to the procedure.

If every module is saved successfully, the procedure returns true. If any module cannot be saved because its file is read only, the procedure posts a warning and returns false, but does *not* signal an error. If a module could not be saved for any other reason, the procedure signals an error.

By continuing to process modules even though a read-only module is encountered, the procedure allows you to save selected modules while preserving the original state of others by keeping them in read-only files.

The g2-save-module system procedure accepts the filename extension you specify and writes the file with the specified extension.

Saving the Current Workspace Layout

You can save the current workspace layout by specifying a g2-window object as the third argument of this procedure. However, while you can name a g2-window, you cannot cause that name to persist through a KB save and reload operation. Therefore, if you name and use a g2-window name as an argument to this procedure, the name will not exist after reloading the KB. The unique properties of G2 window items are described in G2-Windows in the *G2 Reference Manual*.

Caution

G2 enters a wait state during execution of this procedure, which allows other processing that can change the G2 context asynchronously. Be sure to revalidate the context as needed after the procedure returns.

If asynchronous processing during a wait state changes a module while the module is being saved, different parts of the resulting file will reflect the state of the module at different times.

Example

The next example uses the g2-save-module system procedure, specifying that the file progress display should appear:

```
save-module(win: class g2-window)
return-value: truth-value;
begin
    return-value =
        call g2-save-module("/home/kmt/kbs/support-module.kb",
                                    true, win, the symbol support-module, true)
end
```

Notice these two points:

- The g2-window designation, this window, cannot be used in a procedure because it requires a workstation context. The this window designation can be passed to a procedure by, for example, an action-button: start save-module(this window). Providing a window as an argument saves the workspace layout of that window.
- The statement the symbol precedes the module name.

g2-save-module-without-other-processing

Saves a module from the current KB without interruption, optionally with all required modules.

Synopsis

g2-save-module-without-other-processing
(*pathname*: text, *file-progress-display*: truth-value, *win-or-none*: item-or-value, *module*: symbol, *include-required-modules*: truth-value)
-> *module-saved*: truth-value

Argument	Description
pathname	The full path and file name of the file where the module is to be saved.
file-progress-display	Specifies whether to show the file-progress display while the module is being saved.
win-or-none	Enter a g2-window object to save the existing workspace layout of a window or none if you do not wish to save the current layout. For additional information about this argument, see Saving the Current Workspace Layout.
module	The name of the module in your current KB that you wish to save programmatically.
include-required-modules	Specifies whether to save any required modules. Specifying this argument as true saves the module with any modules it requires.

Return Value	Description
module-saved	A truth-value that is true if every module was saved successfully, or false if any module was not saved because its file is read-only.

Description

This procedure saves a module in a specified file, and optionally saves all modules that the named module requires. It saves your KB by first pausing G2, saving your KB, and then resuming G2. When saving required modules, the procedure saves each one in the directory specified in the first argument to the procedure.

If every module is saved successfully, the procedure returns true. If any module cannot be saved because its file is read only, the procedure posts a warning and returns false, but does *not* signal an error. If a module cannot be saved for any other reason, the procedure signals an error.

By continuing to save your KB even though it has read-only modules, the procedure allows you to save selected modules while preserving the state of others by keeping them in read-only files.

g2-reclaim-save-module-memory

Reclaims the memory consumed by g2-save-module.

Synopsis

g2-reclaim-save-module-memory
    ()

Description

It's observed that g2-save-module consumes memory in each call. The reason for memory consumptions are as follows:

- G2 Server doesn't release Operator Logbook messages.
- G2 Server doesn't delete workspaces for Operator Logbook.
- Initially rules will *not* be executed. If the KB contains a user-defined procedure named warmboot, which has no arguments, G2 will execute that procedure.
- After the warmboot procedure has executed and completes, all scanned rules start to execute.

The memory consumed by above two reasons can be reclaimed by g2-reclaim-save-module-memory.

g2-snapshot

Saves a snapshot of a running KB.

Synopsis

g2-snapshot
(*pathname*: text, *file-progress-display*: truth-value, *win-or-none*: item-or-value)
-> *snapshot-saved*: truth-value

Argument	Description
pathname	The name of the snapshot in your current KB that you wish to save programmatically.
file-progress-display	Specifies whether to show the file-progress display while the snapshot is being saved.
win-or-none	Enter a g2-window object to save the existing workspace layout of a window or none if you do not wish to save the current layout. For additional information about this argument, see Saving the Current Workspace Layout.

Return Value	Description
snapshot-saved	A truth value that is true if the snapshot is saved successfully or false if it is not.

Description

Lets you save a snapshot of a running KB, which captures all of its transient data as of the moment you invoke the procedure. Once G2 saves a snapshot file, you can use load it with g2-warmboot-kb.

Creating a Snapshot

When you invoke g2-snapshot, the system procedure:

1. Opens a temporary snapshot file and writes a short preamble to it.
2. Writes all of the current KB's class definitions to the temporary snapshot file.

3 Permits other KB processing to take place, but continues to write all remaining items into the temporary snapshot file.

   If KB processing includes changing an existing item, g2-snapshot first writes the item to the temporary snapshot file, and then completes the item change.

4 Completes writing all KB data to the temporary snapshot file, then closes the file.

5 Renames the temporary snapshot file to the name that you specified.

If you *pause*, *reset*, or *restart* the KB after invoking g2-snapshot, the system procedure completes writing the snapshot file, without interruption, before allowing any further KB processing. For a KB whose contents are constantly changing, and which therefore takes a long time to save with the system procedure, pausing the KB lets g2-snapshot complete without interrupts.

Snapshot File Contents

A KB snapshot file records:

- All information necessary to present the KB as if it had been reset at the time of the snapshot, including information necessary to undo changes that are normally undone when a KB is reset
- All transient items
- The current values, collection times, expiration times, and histories of all variables and parameters when present
- The simulation values and histories of all variables when present
- The activation status of all KB workspaces
- All instances of dynamic relations
- The contents of all lists and arrays

The g2-snapshot procedure saves the state of the entire KB in a single file, even when the current KB is modularized. When you later warmboot with a single snapshot file of a modularized KB, the KB includes all required modules.

Caution

G2 enters a wait state during execution of this procedure, which allows other processing that can asynchronously change the G2 context. Be sure to revalidate the context as needed after the procedure returns.

Snapshot File Errors

If an error occurs before g2-snapshot renames the temporary snapshot file, the temporary file is deleted. Deleting the file prevents overwriting a successfully saved snapshot file from a previous save. Two reasons for not completing the snapshot are:

- Aborting the system procedure.
- Inability to open, write to, or close the snapshot file because of a system-level constraint, such as lack of write privileges, or another problem, such as a network disconnect.

g2-snapshot-without-other-processing

Saves a snapshot of a running KB without interruption.

Synopsis

g2-snapshot-without-other-processing
(*pathname*: text, *file-progress-display*: truth-value, *win-or-none*: item-or-value)
-> *snapshot-saved*: truth-value

Argument	Description
pathname	The pathname of the file where you wish G2 to save a snapshot of your current KB.
file-progress-display	Specifies whether to show the file-progress display while the snapshot is being saved.
win-or-none	Enter a g2-window object to save the existing workspace layout of a window or none if you do not wish to save the current layout. For additional information about this argument, see Saving the Current Workspace Layout.

Return Value	Description
snapshot-saved	A truth value that is true if the snapshot is saved successfully or false if it is not.

Description

Lets you save a snapshot of a running KB, which captures all of its transient and permanent data as of the moment you invoke the procedure. It saves your KB by first pausing G2, saving your KB, and then resuming G2. Once G2 saves a snapshot file, you can load it with the g2-warmboot-kb system procedure.

Snapshot File Contents

A KB snapshot file records:

- All permanent and transient items, including any transient attribute changes
- All information necessary to undo transient changes that are normally undone when a KB is reset, so that resetting G2 can function correctly

- The current values, collection times, expiration times, and histories of all variables and parameters when present
- The simulation values and histories of all variables when present
- The activation status of all KB workspaces
- All instances of dynamic relations
- The contents of all lists and arrays

This procedure saves the state of the entire KB in a single file, even when the current KB is modularized. When you later warmboot with a single snapshot file of a modularized KB, the KB includes all required modules.

g2-warmboot-kb

Loads and starts a KB saved previously with the g2-snapshot system procedure.

Synopsis

g2-warmboot-kb
(*pathname-string*: text, *run-from-snapshot-time-fast:* truth-value)
-> *warmbooted*: truth-value

Argument	Description
pathname-string	The name of a KB that was previously saved using the g2-snapshot system procedure and which you now wish to warmboot.
run-from-snapshot-time-fast	Determines whether the KB that you warmboot is run in catch-up mode from its previous save time after it is loaded with this system procedure. If this argument is true, the KB starts with the G2 clock set to the time that you saved the KB. G2 then runs the KB in as fast as possible mode. If this argument is false, the KB starts with the G2 clock set to the current time.

Return Value	Description
warmbooted	A truth-value that returns true if the KB specified by pathname is warmbooted successfully or false if it is not.

Description

This system procedure provides programmatic access to the Load KB menu choice with the warmbooting afterwards option. The KB you are loading must previously have been saved with the g2-snapshot system procedure.

When a KB is warmbooted with the g2-warmboot-kb system procedure, several things occur:

- The KB is started upon load completion.
- Except for information necessary to present the KB as if it had been reset at the snapshot checkpoint, all runtime data will be present.
- Initially rules will *not* be executed. If the KB contains a user-defined procedure named warmboot, which has no arguments, G2 will execute that procedure.
- After the warmboot procedure has executed and completes, all scanned rules start to execute.

If there is no warmboot procedure, scanned rules start to execute immediately after KB loading and starting.

Warmbooting With Catch-Up

By specifying the second argument of this procedure as true, you can run the KB in catch-up mode from the last time it was saved. Executing the system procedure this way to initialize the KB snapshot file causes G2 to set:

- The scheduler's internal current time setting to the current time saved in the KB snapshot file.
- The scheduler-mode attribute of the Time Parameters system table to as fast as possible.

After warmbooting a KB in this manner, if you do not wish the KB to continue running in as fast as possible mode, you should reset the scheduler-mode attribute to the value real-time as soon as the Scheduler's current time becomes equal to the current real time.

Warmbooting with GFR

In general, a warmbooted KB should use GFR capabilities to insure that the warmboot has the desired effect. For further information, see Knowledge Bases in the *G2 Reference Manual* and the *G2 Foundation Resources User's Guide*.

Example

An example of using the g2-warmboot-kb system procedure, specifying that the run-from-snapshot-time-as-fast-as-possible argument is true, is:

```
test-warnboot( )
warmbooted: truth-value;
begin
    warmbooted = call g2-warmboot-kb("/home/ghw/kbs/snap-test.kb", true);
    if warmbooted then post "Warmboot has succeeded."
        else post "Warmboot did not complete."
end
```

16

List, Array, and Sequence Operations

Describes procedures for getting list, array, and sequence element positions.

Introduction

The list, array, and sequence system procedures return the element index of a given item-or-value in a list, array, or sequence. These procedures provide a fast alternative to iterative searches.

g2-get-position-of-element-in-array

Returns the index of a particular array element if it exists.

Synopsis

g2-get-position-of-element-in-array
(*element*: item-or-value, *array*: class g2-array,
-> *element-index*: integer

Argument	Description
element	The element to find in the array. This can be an item or any value, except a sequence or structure.
array	The target array in which to locate the element.

Return Value	Description
element-index	The element index of the element.

Description

This procedure returns an integer indicating the location of the first occurrence of the element indicated by the *element* argument. If the element is not found in the array, the procedure returns -1.

g2-get-position-of-element-in-list

Returns the index of a particular list element if it exists.

Synopsis

g2-get-position-of-element-in-list
(*element*: item-or-value, *list*: class g2-list,
-> *element-index*: integer

Argument	Description
element	The element to find in the list. This can be an item or any value, except a sequence or structure.
list	The target list in which to locate the element.

Return Value	Description
element-index	The element index of the element.

Description

This procedure returns an integer indicating the location of the first occurrence of the element indicated by the *element* argument. If the element is not a member of the list, the procedure returns -1.

g2-get-position-of-element-in-sequence

Returns the position of an item-or-value in a sequence.

Synopsis

g2-get-position-of-element-in-sequence
(*element*: item-or-value, *sequence*: sequence)
-> *item-position*: integer

Attribute	Description
element	The item or value to obtain a position for within *sequence*.
sequence	The sequence to search for *element*.

Return Value	Description
item-position	The numerical position of the first instance of the item or value, if it is found, or -1, if it is not.

Description

This procedure returns the position of a given item-or-value in a sequence. If the element appears more than once, the position of its first appearance is returned.

17

Math Operations

Describes dense G2 arrays, sparse G2 arrays, G2 matrixes, and the system procedures that create, manipulate, and access them. Also describes system procedures for generating random numbers from a random seed.

Introduction

Use the matrix system procedures for handling:

- Aggregator arrays
- Dense arrays and matrices
- Sparse arrays

Use the random number generator system procedures for generating random numbers from a random seed.

Representing Matrices

A **matrix** is an item-array whose **elements** are quantity-arrays. Each quantity-array represents one row of the matrix.

G2 does not support a matrix class. However, you can assemble data structures that serve as matrices in your KB by creating item arrays and quantity arrays.

Representing Arrays

A **dense array** is a quantity array that can include elements whose value is 0.

A **sparse array** is an alternate way of expressing a dense array. A sparse array consists of two separate arrays:

- A quantity array, which holds only the non-zero elements of the corresponding dense array. This array is called the **value array**.
- An integer-array, which holds the index in the dense array of each non-zero element. This array is called the **index array**.

For example, a dense array can contain the following elements:

(4.0, 0.0, 7.0, 0.0, 9.2, 0.0, 0.0, 0.0, 0.0)

The sparse array that corresponds to this dense array consists of a value array containing the non-zero elements of the dense array:

(4.0, 7.0, 9.2)

and an index array containing the index of each non-zero element:

(0, 2, 4)

Creating Objects Referenced by Procedures

All arguments of matrix system procedures must refer to existing objects. The procedures do not create the objects referenced by their arguments.

For information about how to create arrays and matrices, see Lists and Arrays in the *G2 Reference Manual*.

Performing Operations on Matrices and Arrays

You can use system procedures to perform:

- Aggregator operations on dense or sparse arrays
- Operations on dense arrays and matrices
- Operations on sparse arrays

Random Number Generator

Use the random number generator system procedures for pseudo random number generation in G2. These procedures provide:

- Improved convenience in repeating the random number sequence.
- User access and control over the seed for the random number generator. For example, if you want repeatability of the random number sequence over different invocations of G2.

Note These system procedures affect the behavior of the random function.

Aggregator Array Operations

Use these system procedures to perform aggregator array operations on dense or sparse arrays. The following procedures accept a single quantity-array as an argument, which can be used on dense arrays or used to represent sparse arrays:

g2-array-max

g2-array-min

g2-array-sum

g2-array-sum-abs

g2-array-max

Returns the maximum value in the array.

Synopsis

g2-array-max
(*value-array*: class quantity-array)
-> *largest-element*: quantity, *index*: integer

Argument	Description
value-array	The array whose maximum value is returned.

Return Value	Description
largest-element	The largest (maximum) value in the array.
index	The index of the maximum value.

Example

```
max-value = call g2-array-max(array-1)
```

g2-array-min

Returns the minimum (smallest) value in the array.

Synopsis

g2-array-min
(*value-array*: class quantity-array)
-> *smallest-element*: quantity, *index*: integer

Argument	Description
value-array	The array whose minimum value is returned.

Return Value	Description
smallest-element	A quantity of the smallest element value in the array.
index	The index of the minimum value.

Example

```
minimum-value = call g2-array-min(array-1);
```

g2-array-sum

Returns the sum of all the elements in the array.

Synopsis

g2-array-sum
(*value-array*: class quantity-array)
-> *sum-of-elements*: quantity

Argument	Description
value-array	The array whose elements are added together.

Return Value	Description
sum-of-elements	A quantity of the sum of values of the elements of the array.

Example

```
array-sum = call g2-array-sum(array-1)
```

g2-array-sum-abs

Returns the sum of the absolute values of all the elements in the array.

Synopsis

g2-array-sum-abs
(*value-array*: class quantity-array)
-> *sum-of-absolute-values*: quantity

Argument	Description
value-array	The array for which the sum of the absolute values of elements is calculated.

Return Value	Description
sum-of-absolute-values	A quantity of the sum of the absolute values of the elements of the array.

Example

```
array-abs-sum = call g2-array-sum-abs (array-1)
```

Dense Array and Matrix Operations

A dense array is a quantity-array. G2 supports the following subclasses of quantity-array: integer-array and float-array. You can define subclasses of these G2 classes. For more information about arrays, see Lists and Arrays in the *G2 Reference Manual*.

For an argument declared as a g2-array, you can specify either a quantity-array or a matrix. However, within the same procedure call, you cannot specify quantity-arrays for some g2-array arguments and matrices for other g2-array arguments. The only exception to this rule is the procedure g2-matrix-multiply, which can accept a mixture of quantity-arrays and matrices as arguments.

The following procedures can take non-square matrices or square matrices as arguments: g2-array-add, g2-array-subtract, g2-array-equal, g2-array-copy, g2-array-transpose, g2-get-matrix-dimensions, and g2-matrix-multiply.

G2 supports the following procedures for operations on dense arrays and matrices:

- g2-array-add
- g2-array-copy
- g2-array-copy-elements-to-initial-values
- g2-array-equal
- g2-array-multiply
- g2-array-subtract
- g2-get-matrix-dimensions
- g2-lu-back-substitute
- g2-lu-decompose
- g2-lu-solve
- g2-matrix-multiply
- g2-scalar-multiply
- g2-transpose

g2-array-add

Adds the elements of two arrays and places the resulting sum in a third array.

Note: there's similar system function called vector-add.

Synopsis

g2-array-add
(*array1*: class g2-array, *array2*: class g2-array,*array3*: class g2-array)

Argument	Description
array1	The first G2 array to add.
array2	The second G2 array to add. **Note**: The dimensions of *array1* and *array2* must be identical.
array3	An array containing the sum of *array1* and *array2*.

Description

G2-array-add adds the first two arguments (arrays) and places the result in the third argument. Corresponding elements of *array1* and *array2* are added to produce the elements of *array3*. For example:

(1, 2, 3) + (4, 5, 6) = (5, 7, 9)

For either *array1* or *array2*, you can specify the same array that you specify for *array3*. In this case, the result is written to *array1* or *array2* as well as to *array3*.

Examples

The following example returns the sum of a1 and a2 to s-array:

```
call g2-array-add (a1, a2, s-array);
```

The following example specifies a1 both as one of the arrays to be added, and as the result array (*array-3*):

```
call g2-array-add (a1, a2, a1);
```

Thus, if a1 = (1, 2, 3) and a2 = (3, 4, 5), this call returns (4, 6, 8) to a1.

g2-array-copy

Copies one item or quantity array into a second item or quantity array. The types and dimensions of the arrays must be identical. The copy overwrites any existing values in the destination array.

Synopsis

g2-array-copy
(*array1*: class {item-array | quantity-array},
array2: class {item-array | quantity-array})

Argument	Description
array1	An item or quantity array.
array2	The array into which *array1* is copied.

Note: The types and dimensions of the arrays must be identical.

Example

This example copies a1 into a2:

```
call g2-array-copy (a1, a2)
```

g2-array-copy-elements-to-initial-values

Sets the initial-values attribute of an array to equal the current values of the array.

Synopsis

g2-array-copy-elements-to-initial-values
(*array1*: class g2-array)

Argument	Description
array1	The array whose initial values are set to equal the array's current values.

Description

This procedure makes a permanent change to the initial-values attribute of the specified array. When G2 is reset, the attribute continues to have the value to which it was set by this procedure.

Example

```
update-initial-values (SomeArray: class g2-array)

{Given a g2-array with inital values: (5, the symbol GOLF, "some text", jack)
where jack is an object and the array-length is 4. Click the table to see the initial
values.If the array has been initialized and you click Describe on the array, the
items and values in the initial values should appear in the description.}

begin

change the array-length of SomeArray to 5;
change SomeArray[0] = "Tennis";
change SomeArray[1] = 14;
change SomeArray[2] = jill;{-- another object --}
change SomeArray[3] = the symbol SOCCER;
change SomeArray[4] = "some more text";
{If you click Describe on the array during the wait, the new values will be listed.
Note that the initial values on the table display remain the same.}

post "waiting for 30 seconds";
wait for 30 seconds;
post "resuming";

call g2-array-copy-elements-to-initial-values (SomeArray);

{ If you click Describe on the array after this procedure has finished,
the 5 new values should now appear as the initial values.}

end
```

g2-array-equal

Tests the equality of the element values of two arrays and returns true or false. This procedure does not modify the arguments.

Synopsis

g2-array-equal
(*array1*: class g2-array, *array2*: class g2-array)
-> *equal-elements*: truth-value

Argument	Description
array1	A G2 array.
array2	A G2 array.

Note: The dimensions of *array1* and *array2* must be identical.

Return Value	Description
equal-elements	A truth-value that is true if each element of the first array is equal to the corresponding element of the second array, or false if it is not.

Example

This example returns true or false to array-equality-truth-value, depending on whether a1 and a2 are equal:

```
array-equality-truth-value = call g2-array-equal(a1, a2)
```

g2-array-multiply

Multiplies two quantity arrays and returns a result of type quantity.

Note: there's similar system function called vector-multiply.

Synopsis

g2-array-multiply
(*array1*: class quantity-array, *array2*: class quantity-array)
-> *products-of-elements*: quantity

Argument	Description
array1	A quantity array.
array2	A quantity array.

Note: The dimensions of *array1* and *array2* must be identical.

Return Value	Description
products-of-elements	A quantity of the sum of the products of the corresponding elements in *array1* and *array2*.

Description

G2-array-multiply multiplies each element in the first array by the corresponding element in the second array. It then adds all the products and returns the sum. For example:

(1, 2, 3) * (4, 5, 6) = 4 + 10 + 18 = 32

Example

The following example returns the product of a1 and a2 to array-product-quantity:

```
array-product-quantity = call g2-array-multiply(a1, a2)
```

g2-array-subtract

Subtracts one G2 array from another G2 array and returns the difference in a third G2 array.

Synopsis

g2-array-subtract
(*array1*: class g2-array, *array2*: class g2-array,
array3: class g2-array)

Argument	Description
array1	A G2 array.
array2	The G2 array that is subtracted from *array1*.
	Note: The dimensions of *array1* and *array2* must be identical.
array3	A G2 array containing the resulting difference.

Description

G2-array-subtract subtracts the second argument (*array2*) from the first (*array1*) and puts the result in the third argument (*array3*). If *array3* equals either of the first two arguments, it overwrites that argument.

Elements of *array2* are subtracted from the corresponding elements of *array1* to produce the elements of *array3*. For example:

(4, 5, 6) - (1, 2, 3) = (3, 3, 3)

Example

This example subtracts array2 from array1 and returns the difference to array-diff:

```
call g2-array-subtract (array1, array2, array-diff);
```

g2-get-matrix-dimensions

Accepts a matrix and returns the dimensions in rows and columns.

Synopsis

g2-get-matrix-dimensions
(*matrix*: class item-array)
-> *matrix-rows*: integer, *matrix-columns*: integer

Argument	Description
matrix	Any matrix.

Return Value	Description
matrix-rows	An integer of the number of rows in the matrix.
matrix-columns	An integer of the number of columns in the matrix.

Example

This example returns the number of rows and columns in matrix1 to nrows and ncols:

```
matrix1: class item-array;
nrows, ncols: integer;
. . .
nrows, ncols = call g2-get-matrix-dimensions (matrix1);
```

g2-lu-back-substitute

Solves the set of N linear equations AX=B.

Synopsis

g2-lu-back-substitute
(*a-matrix*: class item-array, *index*: class integer-array,
b-vector: class float-array, *x-vector*: class float-array)

Argument	Description
a-matrix	A square matrix.
index	The permutation array computed by g2-lu-decompose.
b-vector	The right-hand side array.
x-vector	The result.

Description

G2-lu-back-substitute solves the set of N linear equations AX=B. In this case, A is the LU decomposition of the original matrix, determined by g2-lu-decompose.

The result is placed in *x-vector*. If *x-vector* is the same as *b-vector*, the original *b-vector* is destroyed. *A-matrix* and *index* are not modified by this system procedure, so they can be used for successive calls with different right-hand side *b-vectors*.

Use g2-lu-back-substitute with g2-lu-decompose for solving linear equations.

Example

See the example for g2-lu-solve.

g2-lu-decompose

Decomposes matrices using Crout's method with partial pivoting. The *lu-matrix* argument holds the result of the decomposition.

Synopsis

g2-lu-decompose
(*a-matrix*: class item-array, *lu-matrix*: class item-array,
index: class integer-array)

Argument	Description
a-matrix	The square matrix to decompose.
lu-matrix	The result of the decomposition. The *lu-matrix* should be the same size as *A-matrix*; otherwise, an error occurs. For example, if *A-matrix* is 3x3, then initialize *lu-matrix* as a 3x3 matrix, for example, an item-array with three float-arrays in each slot. May be the same as *a-matrix*, in which case, the original *a-matrix* is destroyed.
index	An integer array that is filled to record the row permutation produced by partial pivoting.

Example

See the example for g2-lu-solve.

g2-lu-solve

Calls the g2-lu-decompose and g2-lu-back-substitute procedures. The argument constraints are the same for those procedures.

Synopsis

g2-lu-solve
(*a-matrix*: class item-array, *lu-matrix*: class item-array,
index: class integer-array, *b-vector*: class float-array,
x-vector: class float-array)

Argument	Description
a-matrix	The square matrix to be decomposed.
lu-matrix	The result of the decomposition. May be the same as *a-matrix*, in which case, the original *a-matrix* is destroyed.
index	An integer-array that is filled to record the row permutation effected by partial pivoting. This argument is used internally by the procedure. It is equivalent to the *index* argument of g2-lu-decompose.
b-vector	The right-hand side array.
x-vector	The result. Must be an existing array, all of whose elements initially are 0.

Description

G2-lu-solve returns to *x-vector* the values that solve the equation:

a-matrix * *x-vector* = *b-vector*

where *a-matrix* is a matrix and *x-vector* and *b-vector* are arrays.

X-vector must contain one element for each column of *a-matrix*. *X-vector* and *b-vector* must have the same dimensions.

G2-lu-solve calls g2-lu-decompose and then calls g2-lu-back-substitute. You can solve a matrix as described in the preceding paragraphs by calling these procedures individually, rather than by calling g2-lu-solve. However, for most purposes, you will find it more convenient to call g2-lu-solve.

Example

Example 1: The following call solves the equation *square* * *x* = *b* for *x*:

```
call g2-lu-solve(square, lu-m, index, b, x);
```

where:

square is the following square matrix:

4, 9, 11
5, 3, 18
8, 7, 33

lu-m is an array that holds the result of the decomposition performed by g2-lu-decompose, which is called by g2-lu-solve. This array is used internally by this procedure call.

index is used internally by this procedure call.

b is the following array: (3, 4, 5)

x is the array whose values must solve the following equation:

square	*	x	=	b
4, 9, 11		X_1		3
5, 3, 18		X_2		4
8, 7, 33		X_3		5

This call to g2-lu-solve returns values for X_1, X_2, and X_3 such that:

$4*X_1 + 9*X_2 + 11*X_3 = 3$, and
$5*X_1 + 3*X_2 + 18*X_3 = 4$, and
$8*X_1 + 7*X_2 + 33*X_3 = 5$

Results: $X_1 = 79/44$, $X_2 = -7.0/44$, and $X_3 = -1.0/4$.

Example 2: Suppose that you change the value of b after solving the equation square * x = b for x. The values of x no longer solve the equation, given the new values of b. To solve the equation for x again, you can call g2-lu-solve. However, in this case, it is more efficient to call the following procedures:

```
call g2-lu-decompose (square, decomp-m, index);
call g2-back-substitute (decomp-m, index, b, x);
```

G2-back-substitute returns to x the new values required to solve the equation. These calls save processing time by using intermediate results produced the first time that the equation was solved; g2-lu-solve, by contrast, performs *all* calculations a second time.

g2-matrix-multiply

Multiplies the first two arguments and stores the product in the third argument.

Synopsis

g2-matrix-multiply
(*matrix*: class item-array, *matrix-or-array1*: class item-array or quantity-array,
matrix-or-array2: class g2-array or quantity-array)

Argument	Description
matrix	Specify a matrix (item-array).
matrix-or-array1	Specify a matrix (item-array) or a quantity-array.
matrix-or-array2	Specify a matrix (item-array) or a quantity-array.

Note: *matrix-or-array1* and *matrix-or-array2* must both be the same type — either matrices (item-arrays) or quantity-arrays.

Description

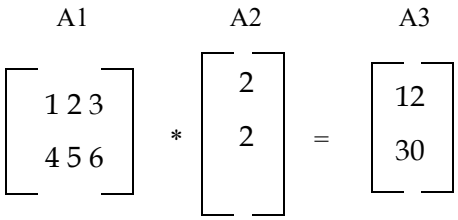
This procedure multiplies *matrix* by *matrix-or-array1* and places the product in *matrix-or-array2*. If *matrix-or-array1* and *matrix-or-array2* are both quantity-arrays, you can specify the same array for both. In this case, the resulting product overwrites *matrix-or-array2*.

Note the following restrictions on the dimensions of *matrix*, *matrix-or-array1* and *matrix-or-array2*:

- The number of columns in *matrix* must equal the number of rows in *matrix-or-array1*.
- The number of rows in *matrix* must equal the number of rows in *matrix-or-array2*.
- The number of columns in *matrix-or-array1* must equal the number of columns in *matrix-or-array2*.

The following examples illustrate how g2-matrix-multiply handles the multiplication of arrays with different dimensions.

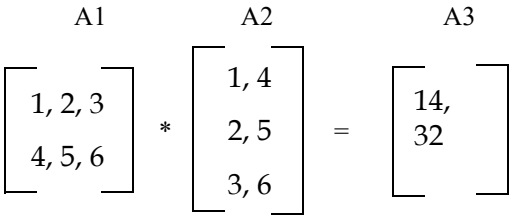
In the following example, each row of the matrix A1 is multiplied by the column in the array A2 to produce the two values in the array A3:



Thus:

(1, 2, 3) * (2, 2, 2) = 2 + 4 + 6 = 12
(4, 5, 6) * (2, 2, 2) = 8 + 10 + 12 = 30

In the following example, each row of the matrix A1 is multiplied by each column of the matrix A2 to produce the products in the matrix A3:



Thus:

(1, 2, 3) * (1, 2, 3) = 1 + 4 + 9 = 14
(1, 2, 3) * (4, 5, 6) = 4 + 10 + 18 = 32
(4, 5, 6) * (1, 2, 3) = 4 + 10 + 18 = 32
(4, 5, 6) * (4, 5, 6) = 16 + 25 + 36 = 77

Example

This example multiplies array1 by array2 and returns the result to product:

```
array1, array2, product: class g2-array:
. . .
call g2-matrix-multiply (array1, array2, product);
```

g2-scalar-multiply

Multiplies a quantity array by a scalar value and returns the result to a second quantity array.

Note: there's similar system function called vector-scalar-multiply.

Synopsis

g2-scalar-multiply
(*array1*: class quantity-array, *x*: quantity,
array2: class quantity-array)

Description

Argument	Description
array1	The quantity array to be multiplied.
x	The scalar value by which *array1* is multiplied.
array2	The quantity array that receives the product of *array1* and *x*.

Note: *array1* and *array2* must be the same length.

Each element in *array1* is multiplied by *x*. For example, if *array1* is (1, 2, 3) and *x* is 5, the result is:

(1, 2, 3) * 5 = (5, 10, 15)

Example

This example multiplies each element of array1 by x and returns the resulting array to array2:

```
call g2-scalar-multiply (array1, x, array2)
```

g2-transpose

Transposes the rows and columns of a matrix.

Synopsis

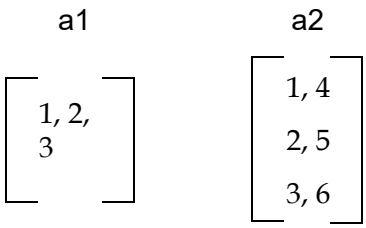
g2-transpose
(*array1*: class item-array, *array2*: class item-array)

Argument	Description
array1	The original matrix.
array2	The transposed matrix.

Description

G2-transpose returns the transposed rows and columns of *array1* to *array2*.

For example, when the rows and columns of matrix a1 are transposed, the result is the matrix a2:



Note the following restrictions on the dimensions of *array1* and *array2*:

- The number of rows in *array1* must equal the number of columns in *array2*.
- The number of columns in *array1* must equal the number of rows in *array2*.

Example

The following example transposes the rows and columns in a1 and returns the result to a2:

```
call g2-transpose (a1, a2)
```

Sparse Array Operations

G2 provides the following system procedures for sparse arrays:

g2-sparse-add

g2-sparse-gather

g2-sparse-get

g2-sparse-multiply

g2-sparse-scatter

g2-sparse-set

g2-sparse-add

Performs the operation X = X + alpha * Y, where X and Y are sparse arrays.

Synopsis

g2-sparse-add
(*value-array-1*: class quantity-array, *index-array-1*: class integer-array,
value-array-2: class quantity-array, *index-array-2*: class integer-array,
full-array: class quantity-array, *alpha*: quantity, *range*: class integer-array,
full-array-initialized: truth-value)

Argument	Description
value-array-1	The value array of the first sparse array.
index-array-1	The index array of the first sparse array.
value-array-2	The value array of the second sparse array.
index-array-2	The index array of the second sparse array.
full-array	A dense array corresponding to the sparse array represented by *value-array-2* and *index-array-2*.
alpha	A value by which to multiply each element of the second sparse array.
range	The range of elements in each sparse array to add. See Description below.

Argument	Description
full-array-initialized	Specify true or false, to specify whether g2-sparse-add adds the two sparse arrays and modifies full-array. If true, and full-array initially contains non-zero elements, g2-sparse-add does not modify full-array. In this case, g2-sparse-add adds the two arrays. If false, g2-sparse-add adds the two arrays and then sets all elements of *full-array* to 0, regardless of the initial values of *full-array*. **Note**: If true, and all elements of *full-array* are initially set to 0, *full-array* is set to the original dense format representation of *value-array-2* and *index-array-2*. In this case, g2-sparse-add does not add the two arrays.

Description

G2-sparse-add performs the operation X = X + *alpha* * Y, where X is the array represented in sparse format by value-array1 and index-array1, and Y is the array represented in sparse format by value-array2 and index-array2. The result is placed in the first sparse array (value-array1 and index-array1).

To perform the addition, g2-sparse-add follows these rules:

- G2-sparse-add adds only those elements of each value array that are within the range specified for that value array. See the description of the *range* argument below.
- If corresponding elements of the two value arrays have identical index values (as listed in their respective index arrays) and each element is within the range specified for its value array, g2-sparse-add adds the two elements and includes the sum as an element of the resulting array. In the resulting array, the sum has the same index value as the two elements that were added.

  Before adding an element of value-array2 to an element of value-array1, g2-sparse-add multiplies the element of value-array2 by *alpha*.
- If a given element of a value array is within the range for that value array but no element in the other value array has the same index value (as listed in its corresponding index array), g2-sparse-add includes the given element in the resulting array, without performing any addition. In the resulting array, the given element keeps its original index value.

If the given element is an element of value-array2, g2-sparse-add multiplies the element by *alpha* before including it in the resulting array.

The range argument is an integer-array containing four elements that specify the range of elements in the two value arrays that can be added:

- The first two elements of range define the range for the first value array, by specifying the indexes in the first value array of the first and last elements to include in the range.
- The last two elements of range define the range for the second value array, by specifying the indexes in the second value array of the first and last elements to include in the range.

For example, to add the first five elements in each value array, specify range as:

```
0, 4, 0, 4
```

To add all the elements of the two value arrays, specify:

```
0, (length of value-array-1 - 1), 0, (length of value-array-2 - 1)
```

Examples

Suppose that g2-sparse-add is called with the following arguments:

```
call g2-sparse-add(value-a, index-a, value-b, index-b, full, 1.0, range, true);
```

and that these arguments have the following values:

```
value-a = (4, 5, 6)
index-a = (0, 1, 2)

value-b = (1, 2)
index-b = (0, 2)
full = (1, 0, 2, 0, 0, 0)

range = (0, 1, 0, 1)

full-array-initialized = true
```

With these arguments, g2-sparse-add returns the following values to value-a:

```
(5, 5, 2)
```

To create this result, g2-sparse-add does the following:

- Adds the first two elements (4 and 1) of the two value arrays. These values arrays have the same index value, 0. The sum is included in the result as the first element (index value 0). Thus, the first element of the result is 5.
- Includes the second element of value-a, 5, in the result. g2-sparse-add does not add this element to the corresponding element of the other value array, because that corresponding element has a different index value.

The index value of the second element of value-a is 1. Thus, the second element of the result is 5.

- Includes the second element of value-b, 2, in the result. g2-sparse-add does not add this element to the corresponding element of the other value array, because the corresponding element has a different index value.

  The index value of the second element of value-b is 2. Thus, the third element of the result is 2.

g2-sparse-gather

Converts a dense array into sparse array format.

Synopsis

g2-sparse-gather
(*value-array*: class quantity-array, *index-array*: class integer-array, *full-array*: class quantity-array)

Argument	Description
value-array	A quantity array containing every non-zero element in *full-array*.
index-array	A quantity array containing the index of every non-zero element of *full-array*.
full-array	The dense array that is converted into a sparse array. All elements of *full-array* are set to 0.

Example

In the following statement:

```
call g2-sparse-gather(v-array, i-array, f-array);
```

v-array and i-array are empty (no elements), and f-array = (0,0,0,0,1).

The statement returns the following argument values:

```
v-array = (1)
i-array = (4)
f-array = (0,0,0,0,0)
```

g2-sparse-get

Returns the quantity value of a specified element of a sparse array.

Synopsis

g2-sparse-get
(*value-array*: class quantity-array, *index-array*: class integer-array,
index: integer)
-> *value-of-index*: quantity

Argument	Description
value-array	The value array component of the sparse array.
index-array	The index array component of the sparse array.
index	The index value, in the original dense array, of the element whose quantity value is returned. This index value can refer to an element whose value is zero as well as to a non-zero element. **Note**: An element whose value is 0 is not included in *value-array*, and its index value is not included in *index-array*.

Return Value	Description
value-of-index	A quantity value of the array element specified by *index*.

Description

G2-sparse-get returns the quantity in the sparse array represented by value-array and index-array whose index in the sparse array is given by index. This is accomplished by indexing value-array with the integer resulting from indexing index-array by index.

Example

The following call to g2-sparse-get returns to element the quantity value of the element whose index is ind:

```
element = call g2-sparse-get(v-array, i-array, ind);
```

Suppose that v-array and i-array represent a dense array with the following values:

```
(0, 1, 0, 2, 0, 0, 3)
```

The following table lists values returned to elements with different values of ind:

Value of ind	Value returned to elements
0	0
1	1
2	0
3	2
4	0
5	0
6	3

g2-sparse-multiply

Multiplies two sparse arrays.

Synopsis

g2-sparse-multiply
(*value-array-1*: class quantity-array, *index-array-1*: class integer-array,
value-array-2: class quantity-array, *index-array-2*: class integer-array,
full-array: class quantity-array, *range*: class integer-array,
full-array-initialized: truth-value)
-> *array-product*: quantity

Argument	Description
value-array-1	The value array of the first sparse array.
index-array-1	The index array of the first sparse array.
value-array-2	The value array of the second sparse array.
index-array-2	The index array of the second sparse array.
full-array	A dense array corresponding to the sparse array represented by *value-array-2* and *index-array-2*.
range	The range of elements in each sparse array that can be multiplied. See Description below.
full-array-initialized	Specify true or false, to specify whether g2-sparse-multiply multiplies the two sparse arrays and modifies *full-array*. If true, and *full-array* initially contains non-zero elements, g2-sparse-multiply does not modify *full-array*. In this case, g2-sparse-multiply multiplies the two arrays. If false, g2-sparse-multiply multiplies the two arrays and then sets all elements of *full-array* to 0, regardless of the initial values of *full-array*. **Note**: If true, and all elements of *full-array* are initially set to 0, *full-array* is set to 0, and quantity is set to 0.

Return Value	Description
array-product	A quantity of the product of the two sparse arrays.

Description

G2-sparse-multiply multiplies the elements of *value-array-1* and *value-array-2* that have the same index values. The index values of elements are listed in the index arrays *index-array-1* and *index-array-2*.

For example, suppose that the first sparse array consists of the following arrays:

value-array-1: (4, 2, 7)
index-array-1: (0, 3, 4)

The second sparse array consists of the following arrays:

value-array-2: (9, 5, 3)
index-array-2: (0, 1, 4)

Only the first and third elements of the two value arrays have the same index values listed for them in *index-array-1* and *index-array-2*. Thus, g2-sparse multiply multiplies only these elements:

(4, 7) * (9, 3) = 36 + 21 = 57

The *range* argument is an integer-array containing four elements that specify the range of elements in each sparse array that can be multiplied:

- The first two elements of *range* specify the indexes in *value-array-1* of the first and last elements that are included in the range.
- The last two elements of *range* specify the indexes in *value-array-2* of the first and last elements that are included in the range.

  For example, to include the first five elements of the two value arrays in the range, specify *range* as:

  0, 4, 0, 4

- To include all the elements of a value array in the range, include the following elements in *range*:

  0, (length of value-array - 1)

  To include all the elements of both sparse arrays in the range, specify *range* as follows:

  0, (length of value-array-1 -1), 0, (length of value-array-2 -1)

Example

The following call to g2-sparse-multiply multiplies two sparse arrays and sets all elements of the full array to 0:

```
sparse-product = call g2-sparse-multiply (varray-1, iarray-1, varray-2,
    iarray-2, farray, range1, false);
```

Because the *full-array-initialized* argument is set to false, this call to g2-sparse-multiply multiplies the two sparse arrays and sets all elements of farray to 0.

Range1 specifies the range of elements in the two sparse arrays that can be multiplied. For example, if the array referenced by the argument range1 is set to (0, 2, 0, 2), all elements of varray-1 and varray-2 are within the range of elements that can be multiplied. (2 = the length of the value arrays minus 1.) Whether the elements are in fact multiplied depends on whether they have the same index values in their index arrays.

g2-sparse-scatter

Converts a sparse array into full dense array format.

Synopsis

g2-sparse-scatter
(*value-array*: class quantity-array, *index-array*: class integer-array,
full-array: class quantity-array, *start-index*: integer, *end-index*: integer)

Argument	Description
value-array	The value array component of the sparse array.
index-array	The index array component of the sparse array.
full-array	The dense array produced by converting the sparse array represented by *value-array* and *index-array*.
start-index	The index in *value-array* of the first element to convert into the dense array.
end-index	The index in *value-array* of the last element to convert into the dense array.

Description

G2-sparse-scatter converts a sparse array represented by *value-array* and *index-array* into a dense array. It returns the dense array to the *full-array* argument.

full-array must have all elements equal to zero when it is passed to this procedure.

start-index is the index of the first element in *value-array* to convert. *end-index* is the index of the last element in *value-array* that is converted. In the dense array, elements have the index values listed in *index-array*.

To convert all elements in *value-array* into a dense array:

- Set *start-index* to 0
- Set *end-index* to (the array-length of value-array - 1).

Example

The following call to g2-sparse-scatter returns the dense array represented by varray and iarray:

```
call g2-sparse-scatter(varray, iarray, farray, 0, 4);
```

Suppose that the sparse array represented by varray and iarray is defined as follows:

```
varray: (2, 4, 6, 3, 1)
iarray: (0, 2, 3, 5, 8)
```

The *start-index* and *end-index* values indicate that the first five elements of varray are to be converted into a dense array. G2-sparse-array uses the values in varray and iarray to produce the following dense array:

```
(2, 0, 4, 6, 0, 3, 0, 0,1)
```

If the *start-index* and *end-index* values are 0 and 1, respectively, g2-sparse-array uses only the first two elements of varray to produce the dense array. In this case, g2-sparse-array returns:

```
(2, 0, 4, 0, 0, 0, 0, 0, 0)
```

g2-sparse-set

Sets a specified element of a sparse array to a specified value.

Synopsis

g2-sparse-set
(*value-array*: class quantity-array, *index-array*: class integer-array,
value-1: quantity, *index*: integer)

Argument	Description
value-array	The value array component of the sparse array.
index-array	The index array component of the sparse array.
value-1	The value to which the specified element of the sparse array is to be set.
index	The index value of the element whose value is to be set. This is the index of the element in the original dense array, as listed for that element in *index-array*.

Description

G2-sparse-set sets the element in the sparse array represented by *value-array* and *index-array* to *value-1* . The index of this element is specified by *index*.

Example

The following call to g2-sparse-set sets the value of the element whose index is ind to the value represented by x:

```
call g2-sparse-set(varray, iarray, x, ind);
```

Suppose that varray and iarray have the following argument values:

```
varray = (1, 2, 3)
iarray = (3, 6, 9)
```

The following table lists the changes to the values of the sparse array specified by different values of x and ind:

x	ind	resulting varray values	iarray
10	3	(10, 2, 3)	(3, 6, 9)
10	0	(1, 2, 3, 10)	(3, 6, 9, 0)
10	8	(1, 2, 3, 10)	(3, 6, 9, 8)

Random Number Generator

Use these system procedures to perform random number generation from a random seed:

g2-repeat-random-function

g2-get-random-seed

g2-set-random-seed

g2-generate-new-random-seed

g2-repeat-random-function

Repeats a sequence of random numbers.

Synopsis

g2-repeat-random-function
    ()

Description

This procedure repeats a sequence of random numbers, that is, with no change in the random seed, without requiring a G2 restart. Compare this system procedure with setting repeat-random-function-on-reset? in the Miscellaneous Parameters system table to yes.

After you call g2-repeat-random-function, the sequence of random numbers the random function returns will be the same as the sequence of numbers it returned since the last time either g2-set-random-seed was called or the KB was (re)started, provided the call to random has the same argument(s).

g2-get-random-seed

Returns the current random seed used to generate random numbers.

Synopsis

g2-get-random-seed
    -> *seed*: integer

Return Value	Description
seed	A non-negative number that is the current random number generator seed.

Description

The random seed is a non-negative integer, which is used to initialize the random number generator.

g2-set-random-seed

Sets a new random seed for generating random numbers.

Synopsis

g2-set-random-seed
(*seed*: integer)

Argument	Description
seed	A non-negative number to use as a random number generator seed.

Description

This procedure enables you to choose the seed, which can be any non-negative integer, used to initialize the random number generator. When you call this system procedure, G2 creates a new random state, using this integer as its seed.

g2-generate-new-random-seed

Generates a new random seed for generating random numbers.

Synopsis

g2-generate-new-random-seed
( )
-> *seed*: integer

Return Value	Description
seed	A non-negative number that can be used as a random number generator seed.

Description

This procedure returns a non-negative integer suitable for initializing the random number generator by calling g2-set-random-seed on this value. This is the type of seed that G2 uses by default when initializing the random number generator. You should not call g2-generate-new-random-seed many times a second if you want to get unique values.

18

Memory Operations

Describes the procedures that help you manage memory.

Introduction

Use the memory system procedures for:

- Measuring and recording item memory
- Item creation tracking

Measuring and Recording Item Memory

These procedures return the number of bytes used by an item and write detailed KB memory statistics to a file:

g2-measure-memory

g2-write-stats

g2-measure-memory

Returns the amount of memory used by any G2 item.

Synopsis

g2-measure-memory
(*item*: class item)
-> *item-memory*: integer,
double-floats: integer,
conses: integer,
vector-slots: integer,
user-vector-slots: integer,
lookup-slots: integer,
frame-vectors: integer,
other-arrays: integer

Argument	Description
item	Any item whose memory usage you wish to obtain.

Return Value	Description
item-memory	An integer giving the amount of total memory used to store *item*.
double-floats	An integer giving the amount of memory used by double-floats to store *item*.
conses	An integer giving the amount of memory used by conses to store *item*. (names, attributes, sequence/structure attribute values, etc.)
vector-slots	An integer giving the amount of memory used by system vector slots to store *item*. (sequence/structure/text/etc. attributes)
user-vector-slots	An integer giving the amount of memory used by user vector slots to store *item*. (user-defined class attributes)
lookup-slots	An integer giving the amount of memory used by lookup slots to store *item*. (attributes with default or symbol values)

Return Value	Description
frame-vectors	An integer giving the amount of memory used by frame vectors to store *item*. (attribute definitions, item configurations)
other-arrays	An integer giving the amount of memory used by other arrays to store *item*. (text attribute values, sequences, structures)

Description

This system procedure returns the amount of memory, in 8-bit bytes, of any item in the KB. Providing a workspace as the argument of this system procedure returns the total memory measurement for every item upon the workspace, including subworkspaces and their corresponding items.

Some memory in use within G2, such as that with the graphics and network services use, is not part of any item. As such, you cannot use the g2-measure-memory system procedure to measure *total* KB memory usage. Use this system procedure to determine item-by-item memory usage. Use **G2 meters** to obtain the *total* amount of memory in use within a KB. Note that similar items might have different amounts of memory allocated, because of the way G2 handles memory.

Caution For very large applications, generating memory statistics can take a long time to execute, effectively requiring you to kill the G2 process. In addition, you should be aware that memory statistics can be very inaccurate, especially in 64-bit versions of G2.

Example

A statement that uses g2-measure-memory to obtain the memory of any workspace that you pass to it, is:

```
memory-tally-integer = call g2-measure-memory(methods-workspace)
```

g2-measure-memory-fields

Returns the meaning (as symbol) of each return values returned by g2-measure-memory.

Synopsis

g2-measure-memory-fields
    ()
    -> *field0*: symbol,
       field1: symbol,
       field2: symbol,
       field3: symbol,
       field4: symbol,
       field5: symbol,
       field6: symbol,
       field7: symbol

Return Value	Description
field0	The symbol sum.
field1	The symbol double-floats.
field2	The symbol conses.
field3	The symbol vector-slots.
field4	The symbol user-vector-slots.
field5	The symbol lookup-slots.
field6	The symbol frame-vectors.
field7	The symbol other-arrays.

Description

This procedure gives you help information to detect the meaning of return values by g2-measure-memory.

Caution The return symbols by this procedure may change in future versions of G2.

g2-write-stats

Allows you to write G2 memory statistics to a file programmatically.

Synopsis

g2-write-stats
(*pathname-string*: text)

Attribute	Description
pathname-string	The pathname of the file to which the stats are written.

Description

This procedure gives you programmatic control over writing G2 memory statistics. It is the programmatic equivalent of selecting Main Menu > Save KB and entering write g2 stats as *pathname*.

You can use memory statistics to help diagnose the source of unbounded memory increases. You may want to produce a series of statistics files for examination by Gensym Customer Support. For additional information, see Memory Management in the *G2 Reference Manual*.

Caution For very large applications, generating memory statistics can take a very long time to execute, effectively requiring you to kill the G2 process. In addition, you should be aware that memory statistics can be very inaccurate.

Item Creation Tracking Procedures

These procedures collectively comprise a tool you can use to monitor item creation:

g2-clear-tracked-items

g2-get-tracked-item-call-sequence

g2-get-tracked-items-call-depth

g2-set-tracked-items-call-depth

g2-start-tracking-items

g2-stop-tracking-items

g2-clear-tracked-items

Instructs G2 to delete item tracking records.

Synopsis

g2-clear-tracked-items
( )

Description

This procedure instructs G2 to clear all item-tracking information. The time to call this procedure is after you have finished examining the Inspect and item describe information from an item-tracking start and stop, and before you restart item tracking. It is essential to call this procedure before restarting item tracking if you do not want the item-tracking records from the previous start and stop to be combined with the new item-tracking records.

g2-get-tracked-item-call-sequence

Returns the procedure calling sequence that leads to the creation of the item argument.

Synopsis

g2-get-tracked-item-call-sequence
(*item*: class item)
-> *procedure-calling-sequence*: sequence

Argument	Description
item	An item created between a call to g2-start-tracking-items and g2-stop-tracking-items.

Return Value	Description
procedure-calling-sequence	An sequence of procedure names, or the empty sequence, sequence( ).

Description

When the tracked-item call depth is set to its default value of 0, the empty sequence is returned; otherwise a sequence of procedure names is returned. You control the depth of procedure-call recording by calling g2-set-tracked-item-call-depth. See the example procedures and return values in the example for g2-set-tracked-items-call-depth.

g2-get-tracked-items-call-depth

Returns the depth to which G2 records the procedures on the calling stack when a tracked item is created.

Synopsis

g2-get-tracked-items-call-depth
( )
-> *depth*: integer

Return Value	Description
depth	The value of the call depth.

Description

This procedure tells you the setting of the current tracked-items call depth. See the previous page for information on setting the depth and what information is available to you when the depth is set to a non-zero value.

g2-set-tracked-items-call-depth

Specifies how deeply G2 should go into the procedure-calling stack to record the sequence of procedure calls that leads to the creation of an item.

Synopsis

g2-set-tracked-item-call-depth
(*depth*: integer)
-> *set-depth*: integer

Argument	Description
depth	An integer value from 0 - 100. The default value is 0.

Return Value	Description
set-depth	The depth value that is set.

Description

When the tracked-item call depth is the default value of 0, G2 does not record any procedure calls. When the depth is 1, G2 records the procedure that contains the item-creation statement; when the depth is 2, G2 also records the procedure that calls the item-creation procedure; and so on to a maximum of 100 procedures.

You can call this procedure at any time; the value you set persists throughout the G2 process until you explicitly change it.

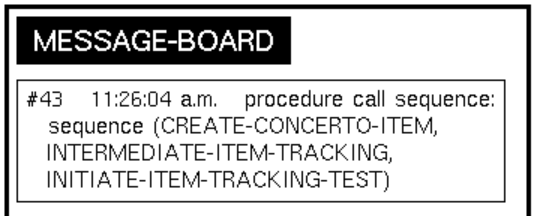
Example

Here are three example procedures that lead to the creation of an item:

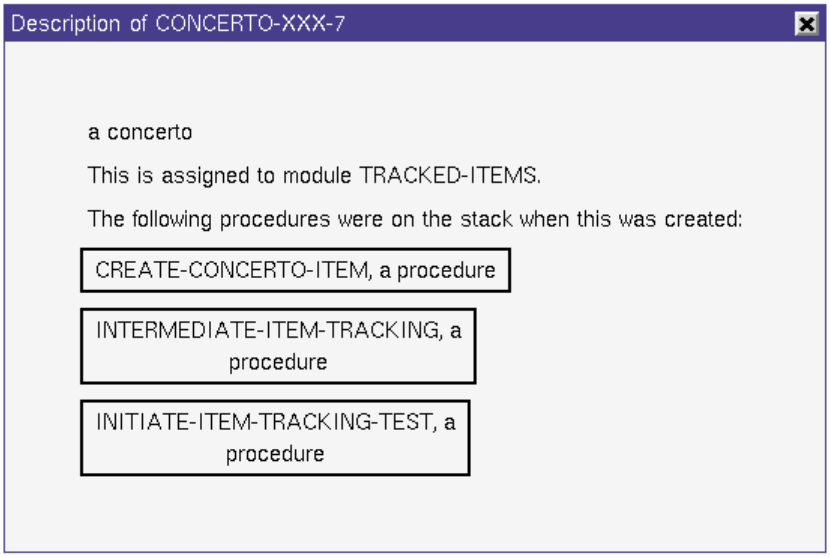
```
initiate-item-tracking-test( )
begin
    call g2-set-tracked-items-call-depth(3);
    call g2-start-tracking-items( );
    call intermediate-item-tracking( );
    call g2-stop-tracking-items( )
end

intermediate-item-tracking( )
begin
    call create-concerto-item( )
end
```

```
create-concerto-item( )
C: class concerto;
procedure-call-sequence: sequence;
begin
     create a concerto C;
     transfer C to test-workspace;
     procedure-call-sequence = call g2-get-tracked-item-call-sequence(C);
     post "procedure call sequence: [procedure-call-sequence]"
end
```

The return value from g2-get-tracked-item-call-sequence is a sequence of three procedure names as shown in the posted message below. G2 returns the empty sequence, sequence( ), when the tracked-item depth is 0.



Until you call g2-clear-tracked-items, the describe menu choice on a tracked item displays procedure-calling sequence information. For example:



g2-start-tracking-items

Instructs G2 to begin keeping a record of every item creation, and to record the procedures that are on the calling stack at the time of creation.

Synopsis

g2-start-tracking-items
    ( )

Description

This procedure tells G2 to start keeping item-creation records, and to continue doing so until you call g2-stop-tracking-items.

The system procedure g2-set-tracked-item-call-depth allows you to specify how deeply G2 should go into the calling stack when recording the procedure-calling sequence that leads to an item's creation.

g2-stop-tracking-items

Instructs G2 to stop keeping records of item creation, and to enable Inspect grammar for displaying the items that have been created, but not deleted, since the *last* call to g2-start-tracking-items.

Synopsis

g2-stop-tracking-items
( )

Description

This procedure tells G2 to stop keeping records of item creation. The first call to this procedure also adds a filter-grammar phrase to the Inspect facility which allows you to execute this command:

show on a workspace every *class-name* in the tracked item set

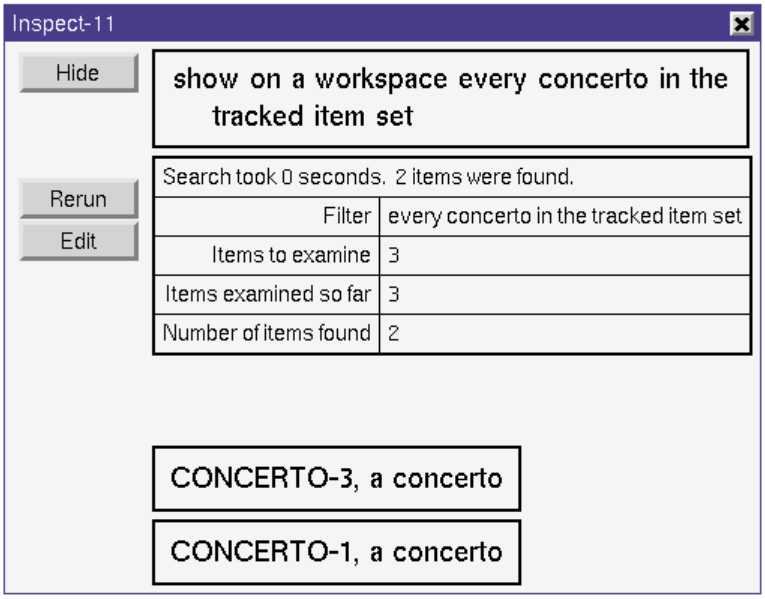
This Inspect command displays short representations of all the instances of *class-name* that have been created between a call to g2-start-tracking-items and g2-stop-tracking-items, but which have not been subsequently deleted. You can also display a workspace of the tracked-item procedure-call sequence by selecting the describe menu option from the item's short representation or from the item's icon.

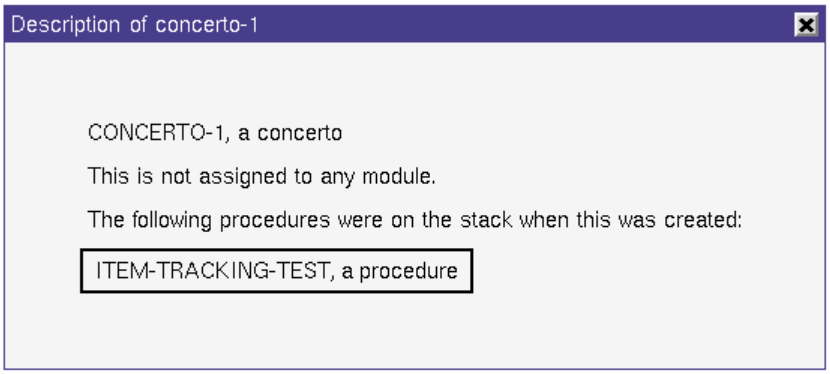
You should execute the Inspect command and describe menu choice before you clear the records with a call to g2-clear-tracked-items. It is, however, essential to call g2-clear-tracked-items before restarting item tracking, otherwise the item-tracking records for each start and stop are combined.

Note The item-tracking filter grammar continues to be available throughout the G2 session; but unless calls to the item tracking procedures have been sequenced as described in these release notes, the value of the number-of-items-found attribute is not meaningful.

Example

```
item-tracking-test( )
counter: integer;
C: class concerto;
begin
     { Create a concerto.  It is not tracked. }
     create a concerto C;
     transfer C to test-workspace;

     { Start tracking item creation. }
     call g2-start-tracking-items( );

     { Create 4 concertos. }
     for counter = 0 to 3
          do
               create a concerto C;
               conclude that the names of  C = symbol("concerto-[counter]")
          end;

     { Delete 2 of the tracked concertos. }
     delete concerto-0;
     delete concerto-2;

     { Stop tracking item creation. }
     call g2-stop-tracking-items( )
end
```


Inspect-11
Hide
show on a workspace every concerto in the tracked item set
Rerun
Edit
Search took 0 seconds. 2 items were found.
Filter
every concerto in the tracked item set
Items to examine
3
Items examined so far
3
Number of items found
2
CONCERTO-3, a concerto
CONCERTO-1, a concerto


Description of concerto-1
CONCERTO-1, a concerto
This is not assigned to any module.
The following procedures were on the stack when this was created:
ITEM-TRACKING-TEST, a procedure

19

Move and Transfer Operations

Describes procedures that gather information about one or more objects for the purpose of moving or transferring the group.

Introduction

Use the move and transfer system procedures to performing the programmatic equivalents of the interactive Operate on Area move and transfer operations.

g2-align-items

Aligns a set of items.

Synopsis

g2-align-items
(*items*: class item-list, *operation*: symbol)

Arguments	Description
items	The list of items to align. Use g2-get-items-in-area to obtain such a list.
operation	The operation to perform. The options are: left, right, top, bottom, left/right-center, and top/bottom-center.

Description

This procedure is the programmatic equivalent of selecting a group of items and right clicking on one to get a popup menu, then choosing align to align the items.

To specify the slash character in a symbol (/), you must use the @ escape character, for example, left@/right-center.

For example, this call aligns the left-right centers of the items in item-list-1:

```
item-list-1: class item-list;

call g2-align-items(item-list-1, the symbol left@/right-center)
```

g2-clone-and-transfer-items

Copies a set of items and transfers them to another workspace.

Synopsis

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

Arguments	Description
items	The list of items to clone and transfer. Use g2-get-items-in-area to obtain such a list.
wksp-destination	The target workspace.
delta-x	The x workspace coordinate as an offset from the items' original x position.
delta-y	The x workspace coordinate as an offset from the items' original x position.

Return Value	Description
transferred-items	A list of all items that were transferred.

Description

This procedure clones a group of items, including any connections between them and any stubs, and transfers the cloned items to another workspace. Connections to items not cloned are not transferred.

Provide a list of the items to transfer as the first argument. Use the g2-get-items-in-area system procedure to get the list of items.

The returned *transferred-items* list contains all items other than connections and stubs that were successfully transferred. Connections and stubs are not included because G2 can obtain them automatically given the listed items.

This procedure is the programmatic equivalent of selecting a group of items and right clicking on one to get a popup menu, then choosing clone and clicking on a workspace to copy the items to that workspace.

g2-clone-and-transfer-items-to-mouse

Copies a set of items and transfers them to the mouse.

Synopsis

g2-clone-and-transfer-items-to-mouse
(*items*: class item-list, *window-destination*: class g2-window)
-> *transferred-list*: class item-list

Arguments	Description
items	The list of items to clone and transfer to the mouse. Use g2-get-items-in-area to obtain such a list.
window-destination	The target window.

Return Value	Description
transferred-list	A list of all items that were transferred.

Description

This procedure clones a group of items, including any connections between them and any stubs, and transfers the cloned items to the mouse. The mouse must be over the workspace on which the items exist. Connections to items not cloned are not transferred.

Provide a list of the items to transfer as the first argument. Use the g2-get-items-in-area system procedure to get the list of items.

The returned *transferred-items* list contains all items other than connections and stubs that were successfully transferred. Connections and stubs are not included because G2 can obtain them automatically given the listed items.

This procedure is the programmatic equivalent of selecting a group of items and right clicking on one to get a popup menu, then choosing clone to transfer the group of items to the mouse.

The *transferred-items* are not active while they are still attached to the mouse and not on any workspace. Therefore, attempts to reference them may result in an error. To refer to the items while they still attached to the mouse, set the may-refer-to-inactive-items hidden attribute to true.

g2-distribute-items

Uniformly distributes a set of items.

Synopsis

g2-distribute-items
(*items*: class item-list, *operation*: symbol)

Arguments	Description
items	The list of items to distribute. Use g2-get-items-in-area to obtain such a list.
operation	The operation to perform. The options are: horizontally and vertically.

Description

This procedure is the programmatic equivalent of selecting a group of three or more items and right clicking on one to get a popup menu, then choosing distribute to distribute the items.

At least three items are required. The outermost two items are unchanged, and the remaining inner items are positioned between the outermost items such that the space between any two items is constant.

For example, this call distributes the items in item-list-1 horizontally:

```
item-list-1: class item-list;

call g2-distribute-items(item-list-1, the symbol horizontally)
```

g2-get-items-in-area

Obtains the items that exist in a specified rectangular area of a workspace.

Synopsis

g2-get-items-in-area
(*wksp*: class kb-workspace, *left*: integer, *top*: integer,
right: integer, *bottom*: integer, *items*: class item-list)

Arguments	Description
wksp	The workspace upon which the items to select reside.
left and *top*	The integer values designating the x and y workspace coordinates of the top left corner of the bounding box.
right and *bottom*	The integer values designating the x and y workspace coordinates of the bottom right corner of the bounding box.
items	A list to which G2 appends all items except connections within the bounding box.

Description

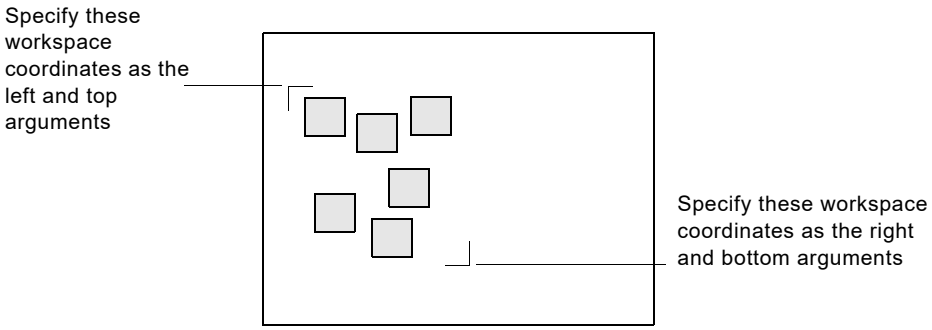
This procedure selects a group of items on a workspace, and appends all but connections and stubs to the existing list *items*. Connections and stubs are not included because G2 can obtain them automatically from the included items. Other group-movement system procedures require such a list of items as their first argument.

You can call this procedure repeatedly with the same items to accumulate a list of the items in more than one area.

Caution To prevent memory leakage, be sure to explicitly delete *items* when you are done with it.

Specifying an Item Bounding Box

You specify the bounding box by providing four integer values that give the top left corner and bottom right corner in workspace coordinates, as follows:



g2-move-items

Moves a group of items from one workspace location to another.

Synopsis

g2-move-items
(*items*: class item-list, *delta-x*: integer, *delta-y*: integer)

Arguments	Description
items	The list of items to move. Use g2-get-items-in-area to obtain such a list.
delta-x	The x workspace coordinate as an offset from the items' original x position.
delta-y	The y workspace coordinate as an offset from the items' original y position.

Description

This procedure moves one or more items, including any connections between them and any stubs, to a new location on the current workspace. Connections to items not moved redraw as needed to maintain their attachments to the moved items.

Provide a list of the items to move as the first argument. Use the g2-get-items-in-area system procedure to get the list of items.

Indicate the new position as x and y workspace coordinates that are the offset from the items' original location. For example, specifying the delta-x and delta-y arguments as 100, 100, respectively, moves the group of items 100 workspace units to the right and 100 workspace units up from their original placement.

This procedure is the programmatic equivalent of selecting a group of items and right clicking on one to get a popup menu, then choosing move and clicking on a workspace to move the items on that workspace.

g2-move-items-to-mouse

Moves a group of items from a location on a workspace to the mouse.

Synopsis

g2-move-items-to-mouse
(*items*: class item-list, *window-destination*: class g2-window)

Arguments	Description
items	The list of items to move to the mouse. Use g2-get-items-in-area to obtain such a list.
window-destination	The target window.

Description

This procedure moves one or more items, including any connections between them and any stubs, to the mouse. The mouse must be over the workspace on which the items exist. Connections to items not moved redraw as needed to maintain their attachments to the moved items.

Provide a list of the items to move as the first argument. Use the g2-get-items-in-area system procedure to get the list of items.

This procedure is the programmatic equivalent of selecting a group of items and right clicking on one to get a popup menu, then choosing move to move the items on the workspace.

This procedure is the programmatic equivalent of selecting a group of items and right clicking on one to get a popup menu, then choosing move to transfer the items to the mouse.

The *transferred-items* are not active while they are still attached to the mouse and not on any workspace. Therefore, attempts to reference them may result in an error. To refer to the items while they still attached to the mouse, set the may-refer-to-inactive-items hidden attribute to true.

g2-transfer-items

Transfers a group of items from one workspace to another.

Synopsis

g2-transfer-items
(*items*: class item-list, *wksp-destination:* class kb-workspace,
delta-x: integer, *delta-y*: integer)

Arguments	Description
items	The list of items to transfer. Use g2-get-items-in-area to obtain such a list.
wksp-destination	The workspace to which to transfer the items.
delta-x	The x workspace coordinate as an offset from the items' original x position.
delta-y	The y workspace coordinate as an offset from the items' original y position.

Description

This procedure transfers one or more items, including any connections between them and any stubs, from one workspace to another. No connection can exist between an item that is transferred and one that is not. If any such connection exists, G2 signals an error and leaves the workspaces unchanged.

Provide a list of the items to transfer as the first argument. To optimize execution of this procedure, set the allow-duplicate-elements attribute of the list to no. Use the g2-get-items-in-area system procedure to get the list of items.

Indicate the position on the destination workspace as x and y workspace coordinates as an offset from the items' original location. For example, specifying the *delta-x* and *delta-y* arguments as 0, 0, respectively, places the group of items on the destination workspace at their original location.

This procedure is the programmatic equivalent of selecting a group of items and right clicking on one to get a popup menu, then choosing transfer and clicking on a workspace to transfer the items to that workspace.

g2-transfer-items-to-mouse

Transfers a group of items from a workspace to the mouse.

Synopsis

g2-transfer-items-to-mouse
(*items*: class item-list, *window-destination*: class g2-window)

Arguments	Description
items	The list of items to transfer to the mouse. Use g2-get-items-in-area to obtain such a list.
window-destination	The target window.

Description

This procedure transfers one or more items, including any connections between them and any stubs, from a workspace to the mouse. The mouse must be over the workspace on which the items exist. No connection can exist between an item that is transferred and one that is not. If any such connection exists, G2 signals an error and leaves the workspaces unchanged.

This procedure is the programmatic equivalent of selecting a group of items and right clicking on one to get a popup menu, then choosing transfer to transfer the items to the mouse.

The *transferred-items* are not active while they are still attached to the mouse and not on any workspace. Therefore, attempts to reference them may result in an error. To refer to the items while they still attached to the mouse, set the may-refer-to-inactive-items hidden attribute to true.

20

Movement Limit Operations

Describes procedures for limiting the movement of items on workspaces.

Introduction

Use the movement limit system procedures to limit an item's movement by the user to a rectangular region. The procedures provide a programmatic equivalent of the corresponding configuration statement:

configure the user interface as follows :
{when | unless} in *x* mode
constrain moving *y* to the rectangle (*left*, *right*, *bottom*, *top*)

where when | unless determines the statement's applicability in a particular mode that *x* indicates, *y* is the item to constrain, and *left*, *right*, *bottom*, and *top* specify the coordinates of an invisible rectangle. Configuring an item with such a statement constrains its movement by the user to the rectangular area.

Note Configuration statements that constrain item movement are applicable only to a user trying to move the item with the mouse. Movement constraint **configurations** do not prevent procedures, methods, or rules from moving the item.

g2-clear-movement-limits

Clears an item's movement limits.

Synopsis

g2-clear-movement-limits
(*item*: class item, *type*: symbol, *mode*: symbol)

Argument	Description
item	The item whose movement limits you want to clear.
type	The type of item movement limit to clear, which can be the symbol when or unless. This argument works in conjunction with the *mode* argument. When clearing an item's movement limit, be sure to specify this argument and the next correctly, as each mode can have a unique set of movement limits.
mode	The mode symbol limiting an item's movement that you want to clear. As the previous argument notes, this argument works in conjunction with the *type* argument.

Description

Use this system procedure to clear the coordinates that limit the user movement of any item (other than a workspace) to a rectangular area. This procedure does not return a value.

The procedure works in conjunction with the other movement-limiting procedures, g2-set-movement-limits and g2-get-movement-limits.

Example

This statement uses g2-clear-movement-limits to clear any movement limits that were set on family-car:

```
call g2-clear-movement-limits(family-car, the symbol when, the symbol user)
```

g2-get-movement-limits

Gets the coordinates that determine an item's movement limits.

Synopsis

g2-get-movement-limits
(*item*: class item, *type*: symbol, *mode*: symbol)
-> *left-boundary*: integer, *right-boundary*: integer,
bottom-boundary: integer, *top-boundary*: integer,
configuration-clause: truth-value

Argument	Description
item	The item whose movement limits you want to retrieve.
type	The type of limit to get for the item, which can be the symbol when or unless. A when value indicates that a certain mode limits movement; an unless value indicates that any mode *except* the one you note limits movement. This argument, which works in conjunction with the mode argument, is identical to the way in which you declare Configuration clauses such as: configure the user interface as follows: when in mode... unless in administrator mode... When getting an item's movement limit, be sure to specify this argument and the next correctly, as each mode can have a unique set of movement limits.
mode	The mode symbol in which to limit the item's movement. As the previous argument notes, this argument works in conjunction with the type argument. As with all Configurations, G2 does not let you restrict administrator mode. Entering the previous argument as when, and this argument as administrator always returns no limits.

Return Value	Description
left-boundary	An integer of the item's left boundary. Returns 0 if no boundary is found.
right-boundary	An integer of the item's right boundary. Returns 0 if no boundary is found.
bottom-boundary	An integer of the item's bottom boundary. Returns 0 if no boundary is found.
top-boundary	An integer of the item's top boundary. Returns 0 if no boundary is found.
configuration-clause	A truth-value that returns true if a configuration clause exists or false if it does not.

Description

Use this system procedure to get the coordinates that limit the user movement of any item (other than a workspace) to a rectangular area.

The procedure searches for a configuration clause corresponding exactly to the values you provide as arguments. For instance, if you enter the type argument as when and the mode argument as supervisor, the procedure searches for a clause that begins:

```
when in supervisor mode:
```

The procedure does not differentiate between whether you added the clause using the g2-set-movement-limits system procedure or another method. If an item has movement limits set, the procedure returns four integer values and a truth-value representing the horizontal and vertical boundaries left, right, bottom, top, and true if it finds a configuration clause. If the procedure does not find a corresponding clause, it returns:

```
0, 0, 0, 0, false
```

The procedure works in conjunction with the other movement-limiting procedures, g2-set-movement-limits and g2-clear-movement-limits.

Example

This procedure illustrates how to get the rectangular coordinates for an item that has a configuration clause corresponding to the system procedure arguments. The procedure displays a message for the operator indicating whether movement limits are set.

```
get-limits(family-car: class car)
current-left, current-right, current-top, current-bottom: integer;
item-has-boundaries: truth-value;
begin
     {get the current boundaries, if any}
     current left, current-right, current-top, current-bottom,
          item-has-boundaries =
               call g2-get-movement-limits(family-car, the symbol when,
                                                    the symbol user);
     if item-has-boundaries
          then post "These are the current movement limits for
               [the name of family-car]: [current-left] [current-right]
               [current-bottom] [current-top]"
          else post "The [the name of car] has no current movement limits."
end
```

g2-set-movement-limits

Sets the coordinates and mode that limit an item's movement upon a kb-workspace.

Synopsis

g2-set-movement-limits
(*item*: class item, *type*: symbol, *mode*: symbol,
left: integer, *right*: integer, *bottom*: integer, *top*: integer)

Argument	Description
item	The item whose movement you want to limit. Using a kb-workspace for this argument does not cause an error, but has no effect.
type	The type of limit to set on the item, which can be the symbol when or unless. A when value indicates that a certain mode limits movement; an unless value indicates that any mode except the one you note limits movement. This argument, which works in conjunction with the *mode* argument, is identical to the way in which you declare configuration clauses such as: configure the user interface as follows: when in mode... unless in administrator mode... By completing this argument and the next, you can set a unique group of movement limits for each different mode.
mode	The mode symbol in which to limit the item's movement. As the previous argument notes, this argument works in conjunction with the *type* argument. As with all configurations, G2 does not let you restrict administrator mode. Entering the previous argument as when, and this argument as administrator causes an error.

Argument	Description
left	An integer value (in workspace units) stating the furthest left position that the item can move horizontally. If this value is *greater* than the *right* value, you cannot move the item horizontally.
right	An integer value (in workspace units) stating the furthest right position that the item can move horizontally. If this value is *less* than the *left* value, you cannot move the item horizontally.
bottom	An integer value (in workspace units) indicating the lowest vertical position that the item can move. If this value is *greater* than the *top* value, the item is unable to move vertically.
top	An integer value (in workspace units) indicating the highest vertical position that the item can move. If this value is *less* than the *bottom* value, the item is unable to move vertically.

Description

Use this system procedure to limit the user movement of any item (other than a workspace) to a rectangular area that you determine. Setting a movement limit on an item adds a corresponding statement to the item's item-configuration attribute, which includes the arguments and values that you pass to the procedure. For example, consider calling the procedure like this:

```
call g2-set-movement-limits (my-object, the symbol when, the symbol user,
    -235, -40, -194, 45)
```

After the procedure returns, the item-configuration attribute of my-object would include this statement:

```
configure the user interface as follows :
    when in mode:
    constrain moving this item to the rectangle (-235, -40, -194, 45)
```

After limiting an item's movement, a procedure can still move it.

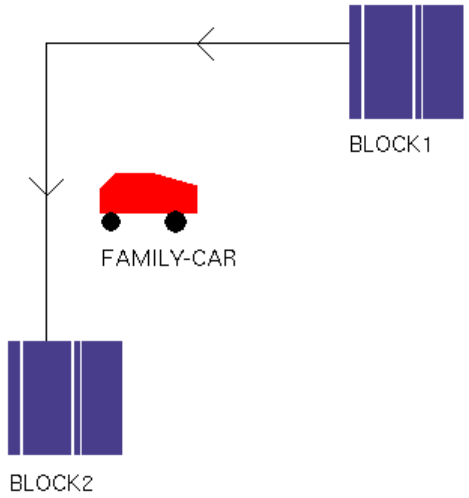
This procedure works in conjunction with the other movement-limiting procedures, g2-get-movement-limits and g2-clear-movement-limits.

Example

This example show a procedure that gets the x and y coordinates of two opposing bounding objects, and then uses those values as the rectangle border for the g2-set-movement-limits system procedure.

Notice that you precede the *type* and *mode* arguments with the symbol statement.

After running this procedure, a KB user in *mode* would be unable to move the family-car beyond the rectangular area:




```
limit-car(a-car: class car)
left, right, bottom, top: integer;
begin
    {get the boundary values from the blocks}
    left = the item-x-position of block1;
    right = the item-x-position of block2;
    top = the item-y-position of block2;
    bottom = the item-y-position of block1;

    {set the boundary limits on the car}
    call g2-set-movement limits(a-car, the symbol when, the symbol user,
                                left, right, bottom, top)
end
```

21

Native Menu System (NMS) API

Describes procedures for programmatically creating native menus in Telewindows.

Introduction

Use the Native Menu System (NMS) API to create and manipulate native menus in Telewindows. The NMS API represents a programmatic way of creating native menus in Telewindows. The other way is to create a graphical menu specification,

using G2 Menu System (GMS), which renders as native menus when viewed through Telewindows.

The NMS API allows you to create menu bars and popup menus. Menus, in turn, consist of menu choices and/or submenus. Menus and menu choices execute a variety of callbacks, which implement the behavior of each menu choice.

Menu bars, menus, and menu choices exists on a per-window basis, that is, for a single Telewindows connection only. Thus, all of the API calls take a g2-window as the last argument.

The Native Menu System (NMS) API allows you to create popup menus on workspace views; however, it does not support menu bars or tool bars on workspace views.

Note The total number of native (NMS) menus and menu items in existence at one time is limited by G2 to around 30,000 per Telewindows connection. This limit comes from a Windows limitation. The exact limits depend on various factors, including which version of Windows you are running, the amount of physical memory installed, and registry settings. The limits are in the neighborhood of 10,000 and includes native windows, menus (but not menu items), and bitmaps, over all processes on the machine. The integer handles used by the native user interface routines in G2 are limited by the largest integer, around 500 million.

The categories of available NMS API procedures are:

- Query and information procedures
- Create and delete procedures
- Menu management procedures
- Get operations
- Set operations
- Toolbar operations

For information and examples of using these procedures, see Windows Menus in the *G2 Reference Manual*.

Callback Procedures

The Native Menu System supports these types of callback procedures:

- Basic callbacks.

  The basic callback is called when the user clicks a choice in an NMS menu or when the user dismisses an NMS menu.

- Extended callbacks.

  The extended callback is also called when the user clicks a choice or dismisses an NMS menu; however, the extended callback provides the text string from edit and combo boxes.

- Selection callbacks.

  Selection callbacks are called when the user highlights a menu choice by dragging the mouse over the choice or by pressing an arrow key, or when the user unhighlights a highlighted menu choice.

- Posting and unposting callbacks.

  The posting callback is called whenever a given menu is posted, that is, when it becomes visible. The unposting callback is called whenever the given menu is unposted, that is, whenever it is no longer visible.

You specify the basic and extended callback procedures:

- When you create a menu bar or menu, using g2-nms-create-menu-bar or g2-nms-create-menu.
- By using the g2-nms-set-callback procedure.

You specify the selection callback procedure by using the g2-nms-set-selection-callback procedure.

You specify the posting callback procedure by using the g2-nms-set-posting-callback procedure and the unposting callback procedure by using the g2-nms-set-unposting-callback procedure.

The procedure argument can be a procedure object, a symbol naming a procedure, or the symbol INHERITED, which uses the callback from the parent menu, if any. NMS signals an error if it cannot find the callback procedure when it tries to call it.

Callbacks are also supported in native GMS.

Basic Callback

The signature of the basic callback procedure is:

my-callback
(*window:* class g2-window, *menu*: integer, *choice*: integer,
activation-path: sequence)

Argument	Description
window	The g2-window for the menu.
menu	The menu handle containing the choice, or the top-level popup menu if the menu was dismissed without choosing.
choice	The handle of the selected menu choice, or 0 if the menu was dismissed without selecting.
activation-path	A sequence of handles of the top-level menu, submenus, and finally the choice, if a choice was made. If a choice was not made, the activation path contains just the top-level menu handle.

Extended Callback

Use the extended callback when you need access to the text string from edit boxes and combo boxes.

The signature of the extended callback is:

my-callback-with-text
(*window*: class g2-window, *menu*: integer, *choice*: integer,
properties: structure)

Argument	Description
window	The g2-window for the menu.

Argument	Description
menu	The menu handle containing the choice, or the top-level popup menu if the menu was dismissed without choosing.
choice	The handle of the menu choice or control that was activated, or 0 if the menu was dismissed without choosing.
properties	A structure with the following syntax: structure (STRING: *text*, PATH: *sequence*) where: • STRING is the current text of the edit box or combo box at the time it was activated, if any. • PATH is the activation path of the choice, which is a sequence of menu choice handles.

Selection Callback

The signature of the selection callback is:

my-selection-callback
(*window*: class g2-window, *menu*: integer, *choice*: integer, *activation-path*: sequence, *status*: truth-value)

Argument	Description
window	The g2-window for the menu.
menu	The menu handle containing the choice.
choice	The handle of the highlighted menu choice.
activation-path	A sequence of handles of the top-level menu, submenus, and finally the choice, if a choice was made.
status	The new state of the choice, where true means the choice was highlighted and false means the choice was unhighlighted.

Posting and Unposting Callbacks

The signature of the posting callback is:

my-callback
(*window*: class g2-window, *menu*: integer, *position*: integer, *activation-path*: sequence)

Argument	Description
window	The g2-window for the menu.
menu	The menu handle containing the choice.
position	The position of the menu in its parent menu.
activation-path	A sequence of handles of the top-level menu, submenus, and finally the choice, if a choice was made.

The signature of the unposting callback is:

my-callback
(*window*: class g2-window, *menu*: integer, *position*: integer, *activation-path*: sequence, *posted*: truth-value)

Argument	Description
window	The g2-window for the menu.
menu	The menu handle containing the choice.
position	The position of the menu in its parent menu.
activation-path	A sequence of handles of the top-level menu, submenus, and finally the choice, if a choice was made.
posted	true when the menu is posted, and false when the menu is unposted.

When called, *menu* is the menu, and *position* is the position of the menu in its parent menu. *Activation-path* is a sequence of menu handles from the root to the given menu. *Posted* is true when the menu is posted and false when it is unposted.

You can use the same callback for posting and unposting a menu. For example:

```
post-unpost-callback(window: class g2-window, menu: integer, pos: integer, path:
sequence, posted: truth-value)
begin
```

```
    if posted then
        change the text of MENU-POST to "Posted menu [menu]."
    else
        change the text of MENU-POST to "Unposted menu [menu].";
end
```

Query and Information Procedures

g2-nms-version
( )
-> *version*: quantity

Returns the current NMS API version number, which is 1.2 in G2 Version 8.0 Rev. 0.

g2-nms-is-supported
(*window:* class g2-window)
-> *menu-bars*: truth-value, *popup-menus*: truth-value

Returns true if the window supports NMS menu bars and popup menus, respectively. This procedure returns true only when Telewindows has been started on a Windows machine, using the `-ui standard` command-line option, which is the default. It returns false when Telewindows has been started with `-ui classic`.

g2-nms-is-menu-bar
(*handle*: integer, *window:* class g2-window)
-> *value*: truth-value

Returns true if the integer handle represents a menu bar. The menu bar does not have to be visible in the window for the procedure to return true. A menu bar consists of one or more menus, which in turn consist of one or more submenus and/or menu choices. See also: g2-nms-set-menu-bar and g2-nms-get-menu-bar.

g2-nms-is-menu
(*handle*: integer, *window:* class g2-window)
-> *value*: truth-value

Returns true if the integer handle represents a menu.

g2-nms-is-choice
(*handle*: integer, *window:* class g2-window)
-> *value*: truth-value

Returns true if the integer handle represents a menu choice.

g2-nms-is-checked
(*choice*: integer, *window:* class g2-window)
-> *value*: truth-value

Returns true if the menu choice is checked; otherwise, returns false.

g2-nms-is-enabled
(*choice*: integer, *window:* class g2-window)
-> *value*: truth-value

Returns true if the menu choice is enabled; otherwise, returns false.

Create and Delete Procedures

g2-nms-create-menu-bar
(*callback*: symbol, *window:* class g2-window)
-> *menu-handle*: integer

Creates an empty menu bar in the window, assuming the window supports NMS menus. Menu choices from menus in this menu bar call the given callback procedure, unless they are overridden by individual choices or submenus. The callback can either be a symbol naming a procedure or an actual procedure object. Use g2-nms-set-callback to override the callback procedure for an individual menu choice.

The menu bar created is a free-floating object. To make the menu bar be the menu bar for the window, you must call nms-set-menu-bar.

This procedure returns the menu bar handle and signals a stack error if it fails.

g2-nms-create-menu
(*callback*: symbol, *window:* class g2-window)
-> *menu-handle*: integer

Creates an empty pulldown or popup menu in the window, assuming the window supports NMS menus. The callback can either be a symbol naming a procedure or an actual procedure object. A callback name of INHERITED causes the menu to inherit the callback from the parent menu. This procedure returns the menu handle and signals a stack error if it fails.

g2-nms-create-submenu
(*window:* class g2-window)
-> *menu-handle*: integer

Creates a submenu in the window, which is identical to creating a menu whose callback is INHERITED. Thus, this procedure is a shorthand for: g2-nms-create-menu (the symbol INHERITED, win)

This procedure returns the menu handle.

g2-nms-delete-menu
(*menu*: integer, *recurse*: boolean, *window:* class g2-window)

Delete the specified menu. If *recurse* is true, then submenus of the menu are also deleted, recursively, to all depths. This procedure signals an error if the specified menu does not exist.

g2-nms-reset
(*window:* class g2-window)

Deletes all menus and their associated menu choices. To show the menus again, you must re-created the menus and their choices. Use g2-nms-dismiss to hide the menus so you can display them again later. Note that resetting all

menus causes the developer menu bar to appear until a new menu bar is created.

g2-nms-dismiss
(*window:* class g2-window)

Dismisses all menus that are currently open in the window, including submenus and popup menus. Note that Windows allows at most one submenu or popup menu to be exposed at a time.

g2-nms-add-submenu
(*menu-bar-or-menu*: integer, *label*: text, *submenu*: integer,
window: class g2-window)
-> *menu-handle*: integer

Appends a menu to the end of a menu bar, thereby creating a pulldown menu, or appends a submenu to the end of a menu, thereby creating a cascading menu. You can only add a submenu to at most one menu bar or menu at a time. The label can contain the ampersand character (&) to underline the following character, thereby creating an access key for displaying the submenu. For example, to define a submenu whose access key is ALT+F, the label would be defined as "&File" which would appear as File in the menu. The label can also contain a tab to right-justify the following characters. The label can have only one of each special character.

This procedure returns the submenu handle. It signals an error if you attempt to add a submenu to more than one menu at a time.

g2-nms-add-choice
(*menu*: integer, *label*: text, *key*: item-or-value, *window:* class g2-window)
-> *choice-handle*: integer

Adds a menu choice to the end of the menu. *Label* is the string for the choice. The label can contain the ampersand character (&) to underline the following character, thereby creating an access key for executing the menu choice. For example, to define a menu choice whose access key is ALT+O, the label would be defined as "&Open" which would appear as Open in the menu. The label can also contain a tab to right-justify the following characters. For example, the menu choice label might include a shortcut key, such as CTRL+O. The label can have only one of each special character. *Key* is a user-defined item or value to associate with the menu choice. Allowable values are: false, true, an integer, a symbol, a string, or an item. This procedure returns the menu choice handle.

g2-nms-add-separator
(*menu*: integer, *window:* class g2-window)
-> *menu-handle*: integer

Adds a separator to the end of the menu. A separator is a horizontal line, which is unselectable. This procedure returns the menu handle.

g2-nms-add-break
(*menu*: integer, *window:* class g2-window)
-> *menu-handle*: integer

Starts a new column in a popup menu or a new line in a menu bar. This procedure returns the menu handle.

g2-nms-add-right-justifier
(*menu*: integer, *window:* class g2-window)
-> *menu-handle*: integer

Causes the following menu choices in a submenu of a menu bar to be right-justified. You can also right-justify individual menu choices when you add them to a menu, using g2-nms-add-menu-choice. This procedure returns the menu handle.

Menu Management Procedures

g2-nms-manage-popup-menu
(*menu*: integer, *x*: integer, *y*: integer, *window:* class g2-window)
-> *menu-handle*: integer

Pops up the menu at the x, y coordinates in the window, assuming the window supports NMS menus. When a choice is made in the popup or the popup is dismissed, the procedure calls the callback procedure. The top-center of the menu is positioned at the x, y coordinates in the window, unless those coordinates would place part of the menu off the screen, in which case the menu pops up as close to the specified position as possible while still being entirely visible. This procedure returns the popup menu handle.

g2-nms-check
(*choice*: integer, *window:* class g2-window)
-> *checked*: truth-value

Places a check mark next to the menu choice. This procedure returns true if the menu choice was previously checked.

g2-nms-uncheck
(*choice*: integer, *window:* class g2-window)
-> *unchecked*: truth-value

Removes a check mark from the menu choice. This procedure returns true if the menu choice was previously unchecked.

g2-nms-radio-check
(*check-handle*: integer, *start-handle*: integer, *end-handle*: integer, *window:* class g2-window)
-> *choice-handle*: integer

Checks one choice in a sequential group of choices in the menu. The check marks display as bullets. This procedure returns the handle of the checked menu choice.

g2-nms-disable
(*menu-or-choice*: integer, *window:* class g2-window)
-> *disabled*: truth-value

Disables (greys out) the menu choice or menu. This procedure returns true if the menu choice was previously disabled.

g2-nms-enable
(*menu-or-choice*: integer, *window:* class g2-window)
-> *enabled*: truth-value

Enables the menu choice or menu. This procedure returns true if the menu choice was previously enabled.

g2-nms-show-menu-bar
(*window:* class g2-window)

Shows the most recently displayed menu bar.

g2-nms-hide-menu-bar
(*window:* class g2-window)

Hides the currently displayed menu bar.

g2-nms-delete-all-menu-choices
(*menu*: integer, *window*: class-g2-window)

Removes all choices in the given *menu*, but not the menu itself. This procedure can be useful when constructing dynamic menus in posting callbacks. For more information, see g2-nms-get-posting-callback and g2-nms-set-posting-callback.

Get Operations

g2-nms-get-built-in-menu
(*menu*: symbol, *win*: class g2-window)
-> *handle*: integer

Returns a copy of one of the following built-in G2 menus:

file
edit
system-tables
view
toolbars
run
run-options
tools
package-preparation
window
help

The choices in the menu function normally, including automatic graying and ungraying by G2. The choices will not call user-defined callback procedures.

g2-nms-get-callback
(*menu-or-choice*: integer, *window:* class g2-window)
-> *callback*: symbol

Returns the name of the callback procedure to be called for the menu or menu choice. Note that the callback might be inherited from a parent menu. Returns the symbol none if no callback is assigned.

g2-nms-get-choices
(*menu*: integer, *window:* class g2-window)
-> *choice-handles*: sequence

Returns a sequence of all the entries in the menu, as handles, where an entry can be a menu choice or a submenu.

g2-nms-get-choice-with-key
(*key:* item-or-value, *menu*: integer, *window:* class g2-window)
-> *choice-handle*: integer

Returns the handle of the menu choice with the given key or none. If more than one choice has the specified key, the procedure arbitrarily returns one of the choices.

g2-nms-get-colors
(*item*: integer, *window:* class g2-window)
-> *foreground*: symbol, *background*: symbol

Returns the foreground and background colors for the menu or menu choice.

g2-nms-get-help
(*menu-or-choice*: integer, *window:* class g2-window)
-> *help*: text

Returns the help string for the menu or menu choice.

g2-nms-get-icon
(*choice*: integer, *window*: class g2-window)
-> *icon:* symbol

Returns the icon from the given *choice*. Returns the symbol none if no icon is assigned.

g2-nms-get-key
(*menu-or-choice*: integer, *window:* class g2-window)
-> *key*: item-or-value

Returns the user-defined key for the menu or menu choice.

g2-nms-get-label
(*menu-or-choice*: integer, *window:* class g2-window)
-> *label*: text

Returns the label for the menu or menu choice.

g2-nms-get-menus
(*window:* class g2-window)
-> *menu-handles*: sequence

Returns a sequence of all the menu handles in the window.

g2-nms-get-menu-bar
(*window:* class g2-window)
-> *menu-handle*: integer

Returns the current menu bar handle. Note that the menu bar may be hidden. A return value of 0 indicates the developer menu bar.

g2-nms-get-posting-callback
(*menu*: integer, *window*: class g2-window)
-> *callback*: symbol

Returns the posting callback for the given *menu*. Returns the symbol none if no callback is assigned.

g2-nms-get-selection-callback
(*menu*: integer, *window*: class g2-window)
-> *callback*: symbol

Returns the selection callback for given menu. Returns the symbol none if no callback is assigned.

g2-nms-get-unposting-callback
(*menu*: integer, *window*: class g2-window)
-> *callback*: symbol

Returns the unposting callback for the given *menu*. Returns the symbol none if no callback is assigned.

Set Operations

g2-nms-set-callback
(*menu-or-choice*: integer, *callback*: item-or-value, *window:* class g2-window)
-> *callback*: symbol

Changes the name of the basic callback procedure to use for all the menu choices in the menu, including all its submenus, or for a particular menu choice. Pass the symbol INHERITED to inherit the callback from the parent menu, if any. The procedure returns the callback.

For the syntax of the callbacks, see Basic Callback and Extended Callback.

g2-nms-set-choice
(*combo-box-handle*: integer, *index*: integer, *window*: class g2-window)

Selects an item in a combo-box displayed in a toolbar, based on its index, where the index of the first item is 0. For example, if a combo-box is displaying the user-mode, and the user-mode changes through some other procedure, use this procedure to update the combo-box to the new value.

g2-nms-set-colors
(*item*: integer, *foreground*: symbol, *background*: symbol,
window: class g2-window)

Sets the foreground and background colors of the menu choice. Pass in the symbol DEFAULT to set or restore the default colors.

g2-nms-set-help
(*menu-or-choice*: integer, *new-help*: text, *window:* class g2-window)

Sets the help string for the menu or menu choice. The help string displays in the status bar when the mouse passes over the menu or menu choice.

g2-nms-set-icon
(*menu*: integer, *icon*: symbol, *options*: structure, *window*: class g2-window)
-> *icon:* symbol

Adds the named *icon* to the left of the *menu* label. The icon occupies the same position as check marks and radio dots. The icon is either the name of a G2 class that defines an icon or one of the pseudo class names listed below. *Options* are currently ignored and should be specified as structure().

If necessary, the icon is scaled down to 16x16 pixels. To avoid scaling, which may affect the appearance, your G2 icons should be smaller than this dimension. To ensure no scaling occurs, your G2 icons should be no larger than 15x15 pixels.

To display only the icon, set the menu choice's label to "".

The following pseudo class names provide built-in icons:

none (this means remove the icon)
gms-cut-icon
gms-copy-icon
gms-paste-icon
gms-undo-icon
gms-redo-icon
gms-delete-icon
gms-file-icon
gms-folder-icon
gms-save-icon
gms-print-preview-icon
gms-properties-icon
gms-help-icon
gms-find-icon
gms-replace-icon
gms-print-icon
gms-gensym-icon (this is the aquamarine gensym logo)
gms-binoculars-icon
gms-save-all-icon

g2-nms-set-key
(*menu-or-choice*: integer, *new-key*: item-or-value, *window:* class g2-window)
-> *menu-or-choice-handle*: integer

Changes the key for a menu choice or menu. *New-key* is a user-defined item or value to associate with the menu choice. Allowable values: false, true, an integer, a symbol, a text string, or an item. This procedure returns the menu or choice handle. See also: g2-nms-add-menu-choice.

g2-nms-set-label
(*choice*: integer, *new-label*: text, *window:* class g2-window)

Changes the label for the menu choice. *New-label* can contain the ampersand character (&) to underline the following character thereby creating an access key. The label can also contain a tab to right-justify the following characters. The label can have only one of each special character. This procedure returns the choice handle. See also: g2-nms-add-menu-choice.

g2-nms-set-menu-bar
(*menu*: integer, *window:* class g2-window)

Sets the menu bar of the window to the given menu. Pass 0 for the handle to revert to the default developer menu bar. See also: g2-nms-create-menu-bar.

g2-nms-set-posting-callback
(*menu*: integer, *callback*: symbol, *window*: class g2-window)
-> *callback*: symbol

Sets the posting callback for the given *menu*. The posting callback is called whenever the given menu is posted, that is, whenever it becomes visible. You use this API to create dynamically updating menus. To remove the callback, specify the symbol none. The procedure returns the callback.

Posting callbacks are inherited from parent menus. Posting callbacks are not allowed on the menu bar itself.

For the syntax of the callback, see Posting and Unposting Callbacks.

g2-nms-set-selection-callback
(*menu*: integer, *callback*: symbol, *window*: class g2-window)
-> *callback*: symbol

Sets the selection callback for the given menu. The selection callback is called when choices in the menu are highlighted and unhighlighted with the mouse or keyboard. To remove the callback, specify the symbol none. The procedure returns the callback.

Selection callbacks are inherited from parent menus. Selection callbacks are not allowed on the menu bar itself.

For the syntax of the callback, see Selection Callback.

g2-nms-set-text
(*edit-or-combo-box-handle*: integer, *new-text*: string, *window*: class g2-window)

Sets the text of an edit-box or a combo-box displayed in a toolbar. For a combo-box, the procedure selects the matching choice if there is one.

g2-nms-set-unposting-callback
(*menu*: integer, *callback*: symbol, *window*: class g2-window)
-> *callback:* symbol

Sets the unposting callback for the given *menu*. The unposting callback is called whenever the given menu is unposted, that is, whenever it is no longer visible. To remove the callback, specify the symbol none.

Unposting callbacks are inherited from parent menus. Unposting callbacks are not allowed on the menu bar itself.

For the syntax of the callback, see Posting and Unposting Callbacks.

Toolbar Operations

Each system procedure takes a window argument and signals an error if the window does not support the function. Each procedure also returns an integer handle that is the toolbar or control.

g2-nms-create-toolbar
(*title*: text, *callback*: symbol, *options*: structure, *win*: class g2-window)
-> *handle*: integer

Creates an empty toolbar in the given window. *options* is a structure with this signature:

```
structure
(dock: symbol,
 neighbor: integer,
 enable-tooltips: truth-value,
 button-style: symbol,
 visible: truth-value)
```

- dock — A symbol that indicates where to dock the toolbar relative to the pane specified by the neighbor attribute. The options are: top, bottom, left, right, float. The default value is top, which docks the toolbar to the top of the overall MDI client window, given the default value for neighbor.
- neighbor — The integer handle of another pane against which to dock. The default value is the overall MDI client window. For example, the following code means dock the toolbar to the right of toolbar1:

  ```
  structure
  (dock: the symbol right, neighbor: toolbar1)
  ```

- enable-tooltips — A *truth-value* that enables the display of tool tips for items on a toolbar. The tooltip is always the same as the item's label (caption). By default, tool tips are disabled.
- button-style — A symbol that sets the button style for all buttons in the toolbar. The options are: automatic, icon-only, caption-only, icon-and-caption (default), or default. automatic omits the label for buttons with icons, other than pulldown menu buttons, where it omits the icon.
- visible — Whether the toolbar is initially visible. Default is true.

g2-nms-create-combo-box
(*label*: text, *choices*: sequence, *options*: structure, *win*: class g2-window)
-> *handle*: integer

Creates a combo box on the given window, where *choices* is a sequence of text strings that provide the choices in the combo box. *options* is a structure with this signature:

```
structure
(initial: text,
key: item-or-value,
width: integer,
drop-down-width: integer)
```

where:

- initial — The initially chosen item, which must be equal to an element in *choices*.
- key — A user-defined item or value to associate with the menu choice. Allowable values are: false, true, an integer, a symbol, a string, or an item.
- width — The width of the combo box, in pixels. The default value is 0, which means a default width that is at least large enough to contain the initial item.
- drop-down-width — The width of the drop-down list of choices, in pixels. The default value is 0, which means a default width that is at least large enough to contain the elements in *choices*.

g2-nms-create-edit-box
(*label*: text, *options*: structure, *win*: class g2-window)
-> *handle*: integer

Creates a single-line edit box on the given window. *options* is a structure with this signature:

```
structure
(initial: text,
key: item-or-value,
width: integer)
```

where:

- initial — The initial contents of the edit box, whose default value is "".
- key — A user-defined item or value to associate with the menu choice. Allowable values are: false, true, an integer, a symbol, a string, or an item.
- width — The width of the edit box, in pixels. The default value is 0, which means a default width that is at least large enough to contain the initial text.

g2-nms-add-control
(menu: integer, *control*: integer, *win*: class g2-window)
-> *handle*: integer

Appends a control to the end of a menu or toolbar, where *control* is the integer handle of the control to add, currently an edit box or combo box.

g2-nms-delete-control
(*control*: integer, *win*: class g2-window)
-> *handle*: integer

Deletes the given control and reclaims its storage.

g2-nms-show-toolbar
(*handle*: value, *win*: class g2-window)

Shows the toolbar specified by *handle* in the specified window, where *handle* can be an integer handle or the name of a toolbar.

g2-nms-hide-toolbar
(*handle*: value, *win*: class g2-window)

Hides the toolbar specified by *handle* in the specified window, where *handle* can be an integer handle or the name of a toolbar.

22

Network Operations

Describes procedures that obtain information about the network and its status, connect to a TCP/IP network host and port, perform input/output operations over a network connection, and ping a network host.

Introduction

G2 allows connecting to and disconnecting from the network programmatically, using TCP/IP sockets.

Use the network operations to:

- Access network information.
- Communicate with network sockets.
- Perform input/output operations over a network connection.
- Pinging and performing a trace route request on a network host.

Network Information

Use thse system procedure to provide programmatic access to the network address on which G2 is listening. G2 listens on this address for connection requests from Telewindows or requests from other processes, such as another G2 or a bridge.

g2-get-host-name

g2-get-network-address-list

g2-get-network-type

g2-get-network-type-given-index

g2-get-port-number-or-name

g2-get-port-number-or-name-given-index

g2-resolve-host-name

g2-get-host-name

Returns the name of the host computer on which G2 is running.

Synopsis

g2-get-host
( )
-> *host-name*: text

Return Value	Description
host-name	A text value naming the host computer on which G2 is running.

Example

A statement that returns the host name of the computer running G2:

```
host-name-text = call g2-get-host-name( )
```

g2-get-network-address-list

Returns a sequence of strings representing the dotted octet notation of all relevant internet addresses of the G2 server machine, both external and internal.

Synopsis

g2-get-network-address-list
( )
-> *ip-addresses*: sequence

Return Value	Description
ip-addresses	A sequence of strings of IP addresses, for example: sequence("192.168.0.2","66.203.92.21")

g2-get-network-type

Returns the type of network that G2 is using.

Synopsis

g2-get-network-type
( )
-> *network-type*: text

Return Value	Description
network-type	A text value indicating the type of network that G2 is using.

Example

A statement that returns the network type for the computer running G2:

```
network-type-text = call g2-get-network-type( )
```

g2-get-network-type-given-index

Returns the current network based on a network type index.

Synopsis

g2-get-network-type-given-index
(*index*: integer)
-> *network*: text

Argument	Description
index	An integer index of 1 or 2.

Return Value	Description
network	A text string of the current network.

Description

Based on the index you provide, this system procedure returns a text value of the current network type that G2 is using. Use this system procedure as a probe for dual-network host systems when you are unsure of how many transport types G2 is using.

Internally, G2 maintains network information within a table. The integer you supply as an argument is an index into that table. The system procedure uses the index in requesting G2 to return information about the network channel referenced at that position.

Example

A statement that returns the G2 network type as a text string based on the index (tcp-ip, in this case):

```
network-type-text = call g2-get-network-type-given-index(1)
```

g2-get-port-number-or-name

Returns the port number or name on which G2 is listening.

Synopsis

g2-get-port-number-or-name
( )
-> *port-number*: text

Return Value	Description
port-number	A text string of the port number or name on which G2 is listening.

Description

Returns the port number or name on which G2 is listening for network connections, such as with G2-to-G2 or Telewindows.

Example

A statement that returns the port name or number of the computer running G2:

```
post-name-text = call g2-get-port-number-or-name( )
```

g2-get-port-number-or-name-given-index

Returns the current port number or name based on a network type index.

Synopsis

g2-get-port-number-or-name-given-index
(*index*: integer)
-> *port-number*: text

Argument	Description
index	An integer index of 1 or 2.

Return Value	Description
port-number	A text string of the current port number or name.

Description

Use this procedure as a probe for dual-network host systems when you do not know how many transport types G2 is using.

Internally, G2 maintains network information within a table. The integer you supply as an argument is an index into that table. The system procedure uses the index in requesting G2 to return information about the network port number or name referenced at that position.

Example

A statement that accepts an index argument to return the port name or number based on the index:

```
port-name-text = call g2-get-port-number-or-name-given-index(2)
```

g2-resolve-host-name

Resolve a hostname and return its primary IPv4 address, with timeout support.

Synopsis

g2-resolve-host-name
(*host*: text, *timeout*: integer)
-> *address*: text

Argument	Description
host	An text string of hostname
timeout	DNS query timeout (in seconds)

Return Value	Description
address	A text string of the returned IPv4 address in dotted numbers (xxx.xxx.xxx.xxx).

Description

Use this procedure as a primary way to resolve any Internet host names. Although it's not necessary to resolve any hostname before making a TCP connection, but using this procedure is the only way to control DNS query timeouts.

Internally, G2 will call an external helper utility called "hostlookup" to do the actual hostname resolve, and the external process will be terminated on timeout. The procedure will signal a HOST-NAME-RESOLUTION-FAILURE error on failures, even when it cannot find the "hostlookup" program from current directory.

Example

A statement that try to resolve a host name and return the result to a text variable, with 5 seconds' timeout:

```
address = call g2-resolve-host-name("www.gensym.com", 5)
```

Network Connection Management

G2 provides classes and system procedures for connecting with network sockets, such as HTTP, and performing input/output operations to read and write data.

The system procedures for interfacing with sockets take as argument and return instances of the g2-socket class. The g2-socket class defines these read-only attributes:

- g2-socket-status — A symbol that indicates that status of the socket. The value is one of these symbols: newly-created, connected, connected-secure, listening, listening-secure, connection-closed, connection-closed-with-unread-data, connection-write-error.
- g2-socket-remote-host — A text that indicates the host name of the socket when available for a connected socket, or no value if the host is a listener socket.

The system procedures for network connection management are:

g2-tcp-accept

g2-tcp-close

g2-tcp-connect

g2-tcp-listen

All system procedures for network socket communication allow other processing.

g2-tcp-accept

Returns a g2-socket that represents an actual connection from an established TCP/IP listener.

Synopsis

g2-tcp-accept
(*socket*: class g2-socket)
-> *socket*: class g2-socket

Argument	Description
socket	The g2-socket that represents the listener.

Return Value	Description
socket	A g2-socket instance. See Network Connection Management for a description.

Description

This system procedure returns a g2-socket instance that is appropriate as the argument to the system procedures used for reading and writing data over a network, as described in Network Reading and Writing.

Generally, a program should establish a listener by calling g2-tcp-listen, then enter a loop accepting connections by calling g2-tcp-accept, starting a new procedure invocation for each connection it received by g2-tcp-accept to handle the I/O.

g2-tcp-close

Closes a TCP/IP connection represented by a g2-socket and deletes the g2-socket.

Synopsis

g2-tcp-close
(*socket*: class g2-socket)

Argument	Description
socket	The g2-socket that represents an established TCP/IP connection. See Network Connection Management for a description.

Description

You call g2-tcp-close on a g2-socket returned by g2-tcp-connect, g2-tcp-accept, or g2-tcp-listen.

The connection may already be disconnected, for example, if the remote end has disconnected. However, because unread data may be buffered from this connection, the system procedure does not delete the g2-socket upon remote end closure, thereby allowing the buffered data to be processed. In this case, the g2-socket-status status of the g2-socket is connection-closed-with-unread-data. You must use this system procedure to remove the socket object from G2 memory.

g2-tcp-connect

Returns a g2-socket representing a TCP/IP network connection.

Synopsis

g2-tcp-connect
(*host*: text, *port*: integer, *options*: structure)
-> *socket*: class g2-socket

Argument	Description
host	A text string representing a network host, for example, "www.gensym.com"
port	An integer, for example, port 80 for HTTP servers.
options	A structure with this syntax: structure (secure: *boolean*) See Description below.

Return Value	Description
socket	A g2-socket instance. See Network Connection Management for a description.

Description

In the *options* structure, secure indicates whether to invoke SSL security on the connection. The default value is false. You can also specify an empty structure.

This system procedure returns a g2-socket instance that is appropriate as the argument to the system procedures used for reading and writing data over a network, as described in Network Reading and Writing.

This system procedure generates an error when various problems occur, for example, when a connection is refused.

g2-tcp-listen

Returns a g2-socket representing a TCP/IP listener.

Synopsis

g2-tcp-listen
(*port*: integer, *options*: structure)
-> *socket*: class g2-socket

Argument	Description
port	An integer, for example, port 80 for HTTP servers.
options	A structure with this syntax: structure (exact: *boolean*, secure: *boolean*, certificate: *text*) See Description below.

Return Value	Description
socket	A g2-socket instance. See Network Connection Management for a description.

Description

This system procedure returns a g2-socket instance that represents a listener, which is appropriate as the argument to g2-tcp-accept.

Note Using port numbers below 1024 on UNIX requires G2 to be running as root, as these are privileged ports.

In the *options* structure:

- exact — Whether to make an exact connection. If true, then if the listener could not be established on the exact port, it generates an error. If false, then if the listener cannot be established on the exact port, it increments the port number and tries again until it finds a port that is available. The default value is false.

- secure — Whether to accept SSL security for clients that connect to this port. This option does not require the client to use SSL; it also accepts insecure connections. The new connection is reported as connected if it is insecure, and connected-secure if it is secure. The default value is false.
- certificate — A string that identifies the SSL certificate to be used if the secure option is set to true. If the `-cert` G2 command line option has been given, it overrides the certificate option in the structure. Also, if another certificate was used to establish security, either for general G2/Telewindows communication or in another g2-tcp-listen call, that certificate is used instead. Thus, only one certificate may be active in a G2 session at one time, and once established, it is used for the remainder of the session.

Network Reading and Writing

In general, the system procedures that perform I/O through sockets uses the same procedure names as the system procedures that perform I/O using streams. However, note that the I/O system procedures for both sockets and streams are implemented as methods rather than as procedures. See Reading and Writing Files.

The system procedures for network connection management are:

- g2-read-block
- g2-socket::g2-read-byte
- g2-socket::g2-read-bytes-as-sequence
- g2-socket::g2-read-bytes-as-text
- g2-socket::g2-read-line
- g2-socket::g2-write-bytes
- g2-socket::g2-write-string

All system procedures for network reading and writing allow other processing.

g2-read-block

Reads a block of text from a connected TCP/IP socket.

Synopsis

g2-read-block
(*socket*: class g2-socket)
-> *block*: text

Argument	Description
socket	A TCP/IP network connection represented by a non-listener g2-socket. See Network Connection Management for a description.

Return Value	Description
block	A text string of the data read.

Description

The g2-read-block system procedure does not have an analogue in file I/O. It simply reads whatever data is presently available or waits for new data and returns whatever block comes in. It returns an empty string when the connection is closed and no more data exists.

g2-socket::g2-read-byte

Reads a byte from a connected TCP/IP socket.

Synopsis

g2-socket::g2-read-byte
(*socket*: class g2-socket)
-> *byte*: integer

Argument	Description
socket	A TCP/IP network connection represented by a non-listener g2-socket. See Network Connection Management for a description.

Return Value	Description
byte	An 8-bit integer byte read from *socket*.

Description

G2-read-byte reads a byte from a connected TCP/IP socket and returns it either as an integer or as -1 if G2 is unable to read it.

g2-socket::g2-read-bytes-as-sequence

Reads a specified number of bytes from a connected TCP/IP socket and returns it as a sequence.

Synopsis

g2-socket::g2-read-bytes-as-sequence
(*socket*: class g2-socket, *n*: integer)
-> *bytes*: sequence

Argument	Description
socket	A TCP/IP network connection represented by a non-listener g2-socket. See Network Connection Management for a description.
n	The number of characters to read.

Return Value	Description
bytes	A sequence of bytes.

Description

G2-read-bytes-as-sequence reads a specified number of bytes from a connected TCP/IP socket and returns it either as a sequence or an empty sequence if G2 is unable to read it.

This version is useful for binary data.

g2-socket::g2-read-bytes-as-text

Reads bytes from a connected TCP/IP socket and returns it as a text.

Synopsis

g2-socket::g2-read-bytes-as-text
(*socket*: class g2-socket, *n*: integer)
-> *bytes*: text

Argument	Description
socket	A TCP/IP network connection represented by a non-listener g2-socket. See Network Connection Management for a description.
n	The number of characters to read.

Return Value	Description
bytes	A text string of the characters read.

Description

G2-read-bytes-as-text reads a specified number of bytes from a connected TCP/IP socket, and returns the result as a text or returns an empty text if the connection is closed.

g2-socket::g2-read-line

Reads a line of text from a connected TCP/IP socket and returns it as a text.

Synopsis

g2-socket::g2-read-line
(*socket*: class g2-socket)
-> *line*: text

Argument	Description
socket	A TCP/IP network connection represented by a non-listener g2-socket. See Network Connection Management for a description.

Return Value	Description
line	A text string of a line read from *socket*.

Description

G2-read-line reads a line from a connected TCP/IP socket.

g2-socket::g2-write-bytes

Writes a sequences of bytes to a TCP/IP network connection represented by a g2-socket.

Synopsis

g2-socket::g2-write-bytes
(*socket*: class g2-socket, *data*: sequence)
-> *success*: truth-value

Argument	Description
socket	A TCP/IP network connection represented by a non-listener g2-socket. See Network Connection Management for a description.
data	The bytes that are written to the socket. The value of *data* is a sequence of 8-bit bytes to be written as binary data and a byte is a value from 0 - 255.

Return Value	Description
success	True if the bytes are written successfully; otherwise, false.

Description

G2-write-bytes writes *data* to *socket*. If the bytes are not written successfully, updates the g2-socket-status to be connection-write-error.

g2-socket::g2-write-string

Writes data to a TCP/IP network connection represented by a g2-socket.

Synopsis

g2-socket::g2-write-string
(*socket*: class g2-socket, *data*: text)
-> *success*: truth-value

Writes data to an existing non-listener connection represented by *socket*, where *data* is a text string.

Argument	Description
socket	A non-listener TCP/IP network connection represented by a non-listener g2-socket. See Network Connection Management for a description.
data	A text string to write to the socket.

Return Value	Description
success	true if the string was written successfully; otherwise, false.

Description

G2-write-string writes *data* to *socket*. If the string is not written successfully, updates the g2-socket-status to be connection-write-error.

Network ICMP Operations

Use these system procedures to ping or make a trace route request on a network host or IP address

g2-ping

Pings a remote host or IP address and returns the statistics.

Synopsis

g2-ping
(*hostname-or-ip*: text, *options*: structure)
-> *info*: structure

Argument	Description
hostname-or-ip	The host name or IP address to ping.
options	Options for pinging the network host or IP address. See Description.

Return Value	Description
info	A structure containing information about the ping request. See Description.

Description

The *options* structure has this syntax:

structure
(packet-size: *integer*,
timeout: *quantity*,
pause-between-ping: *quantity*,
number-of-pings: *integer*,
ttl: *integer*,
progress-procedure: *symbol*,
progress-procedure-user-data: *item-or-value*)

where:

- packet-size is the size of the packet for the ping request. The default is 32 bytes.
- timeout is the minimum timeout period rounded based on the minimum scheduling interval. The default is 10 seconds.
- pause-between-ping is the minimum time, in seconds, for pausing between ping attempts, rounded based on the minimum scheduling interval. The default is 0.

- number-of-pings is the number of ping attempts. The default is 1.
- ttl is is the maximum number of routers through which this packet should travel, which is an integer between 0 and 255. Each time an IP packet goes through a router, its ttl value is decremented by 1.
- progress-procedure is a symbol naming a callback procedure, which is called after each ping is received, where the event is the symbol ping-reply, or after a timeout occurs.
- progress-procedure-user-data is passed as the *data* argument to the progress procedure.

The signature of the progress procedure is:

```
my-ping-progress-callback
    (event: symbol, data: value, user-data: item-or-value)
```

When the *event* is ping-reply, the *data* argument contains a structure with the same syntax as the *info* structure return value to g2-ping.

The *info* structure has this syntax:

```
structure
(hostname-or-ip: text,
packets-transmitted: integer,
packets-received: integer,
packets-lost: integer,
packet-size: integer,
minimum-round-trip-in-s: quantity,
maximum-round-trip-in-s: quantity,
average-round-trip-in-s: quantity,
reply-type: symbol, [none | icmp_echoreply | icmp_timxceed]
last-reply-type: symbol, [none | icmp_echoreply | icmp_timxceed]
last-hostname-or-ip: text,
packets: sequence)
```

where packets is a sequence of structures, one for each ping request, where each structure has this syntax:

```
structure
(hostname-or-ip: text,
reply-type: symbol,
round-trip-in-s: quantity,
ttl: integer)
```

g2-trace-route

Detects the reachability of a remote host or IP address and number of intermediate hops, and returns the statistics.

Synopsis

g2-trace-route
(*hostname-or-ip*: text, *options*: structure)
-> *info*: structure

Argument	Description
hostname-or-ip	The host name or IP address for the request.
options	Options for making the request of the network host or IP address. See Description.

Return Value	Description
info	A structure containing information about the trace route request. See Description.

Description

The *options* structure has this syntax:

structure
(maximum-hops: *integer*,
packet-size: *integer*,
timeout: *integer*,
number-of-pings: *integer*,
progress-procedure: *symbol*,
progress-procedure-user-data: *item-or-value*)

where:

- maximum-hops is the maximum number of hops to detect. The default is 30.
- packet-size is the size of the packet for the request. The default is 32 bytes.
- timeout is the time before the procedure times out. The default is 10 seconds.
- number-of-pings is the number of ping attempts. The default is 1.
- progress-procedure is a symbol naming a callback procedure, which is called at the end of each hop, where the event is the symbol trace-route-hop.

- progress-procedure-user-data is passed as the *data* argument to the progress callback procedure.

The signature of the progress procedure is:

```
my-trace-route-progress-callback
    (event: symbol, data: value, user-data: item-or-value)
```

When the *event* is trace-route-hop, the *data* argument contains a structure with the same syntax as the *info* structure return value to g2-trace-route.

The *info* structure has this syntax:

```
structure
(hostname:-or-ip text,
maximum-hops: integer,
hops: sequence)
```

where hops is a sequence of structures, one for each hop, where each structure has this syntax:

```
structure
(hop: integer,
hostname-or-ip: text,
reply-type: symbol, [none | icmp_echoreply | icmp_timxceed]
packets-transmitted: integer,
packets-received: integer,
packets-lost: integer,
packet-size: integer,
minimum-round-trip-in-s: quantity,
maximum-round-trip-in-s: quantity,
average-round-trip-in-s: quantity)
```

23

Object Passing Operations

Describes procedures that support item- and object-passing in your KB.

Introduction

G2 supports **item passing** and **object passing** using a G2-to-G2 or a G2 Gateway (GSI) interface.

Before passing items to a remote process, you must first register each item to pass, using the object passing system procedures. G2 assigns a **network handle** to each item you register. A handle is an integer value associated with a G2 data interface object.

For more information about item- and object-passing, see G2-to-G2 Interface and G2 Gateway in the *G2 Reference Manual*.

g2-current-remote-interface

Returns the gsi-interface item that can be used to call or start remote procedures in the GSI that invoked the currently running procedure.

Synopsis

g2-current-remote-interface
( )
-> *interface*: class network-interface

Return Value	Description
interface	The gsi-interface item needed to call or start remote procedures.

Description

If the currently running procedure was invoked remotely by GSI, g2-current-remote-interface returns the gsi-interface that is connected to that GSI process. If the currently running procedure was not invoked by GSI, the procedure signals an error.

g2-deregister-on-network

Deregisters an item from network use.

Synopsis

g2-deregister-on-network
(*registered-item-or-handle*: item-or-value, *icp-interface*: class item)
-> *deregistered*: truth-value

Argument	Description
registered-item-or-handle	Use either the previously registered item, or the integer network handle returned by the g2-register-on-network system procedure.
icp-interface	This item must be either a GSI-interface or a G2 data interface object, and refers to the ICP-interface with which the network handle is associated.

Return Value	Description
deregistered	A truth-value that is true if the item is successfully deregistered or false if it is not.

Description

This system procedure deregisters the item from network use by removing the association between it and its handle, and the corresponding ICP-interface, returning true after successfully deregistering the item.

As noted in g2-register-on-network, registering a GSI variable with a GSI-interface as the **Intelligent Communications Protocol** (ICP) interface provides the network handle that GSI uses. When you pass a GSI-variable to g2-deregister-on-network as the item, and GSI-interface as the ICP-interface, GSI data service for the variable stops, since GSI data service is not possible without a network handle to the variable.

If this procedure fails, it signals a stack error.

g2-get-item-from-network-handle

Returns the registered item associated with a network handle.

Synopsis

g2-get-item-from-network-handle
(*network-handle*: integer, *icp-interface*: class item)
-> *registered-item*: class item

Argument	Description
network-handle	The handle originally obtained using the g2-register-item-on-network system procedure.
icp-interface	This item must be either a GSI-interface or a G2 data interface object, and refers to the ICP-interface with which the network handle is associated.

Return Value	Description
registered-item	The registered item associated with a network handle.

Description

This procedure returns the item that is associated with a particular network handle.

If this procedure fails, it signals a stack error. Two possible error symbols are out-of-range-handle and no-such-item-registered.

g2-get-network-handle-from-item

Returns the network handle of any registered item.

Synopsis

g2-get-network-handle-from-item
(*item-to-register*: class item, *icp-interface*: class item)
-> *network-handle*: integer

Argument	Description
item-to-register	The item whose network handle you wish to obtain.
icp-interface	This item must be either a GSI-interface or a G2 data interface object, and refers to the ICP-interface with which the network handle is associated.

Return Value	Description
network-handle	An integer representing the network handle of the registered item.

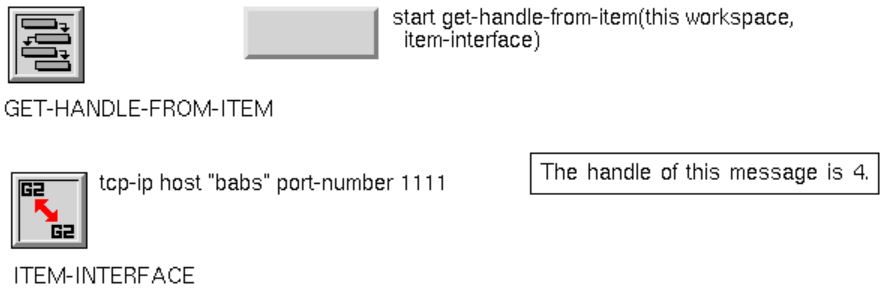
Description

This system procedure returns the network handle of any item. An item that you pass to this system procedure must have been previously registered as a network item either explicitly using the g2-register-on-network system procedure, or implicitly by being used in a remote procedure that declared the item as a handle.

This procedure signals an error if it fails. Two possible error symbols are cannot-register-non-item and no-such-handle.

Example

Here is a section of a workspace that includes a procedure that created a message, registered all the items on the workspace, obtained the handle of the message, and changed the message text to display the message handle number:




```
get-handle-from-item(interface-workspace: class kb-workspace,
                     data-interface-object: class g2-to-g2-data-interface)
HANDLE: integer;
M: class message;
WS-ITEM: class item;
begin
    create a message M;
    transfer M to interface-workspace at (100, -100);
    for WS-ITEM = each item upon interface-workspace
        do
            call g2-register-on-network(WS-ITEM, data-interface-object)
        end
    HANDLE =
        call g2-get-network-handle-from-item(M, data-interface-object);
    change the text of M to "The handle of this message is [HANDLE]"
end
```

g2-register-on-network

Registers any KB item for network use.

Synopsis

g2-register-on-network
(*item-to-register*: class item, *icp-interface*: class item)
-> *network-handle*: integer

Argument	Description
item-to-register	The KB item you are registering to obtain a network handle.
icp-interface	This item must be either a GSI-interface or a G2 data interface object, and refers to the ICP-interface with which the network handle is associated.

Return Value	Description
network-handle	A integer representing the network handle for the item.

Description

This system procedure returns a network handle (an integer) for use during item passing. After network registration, item passing lets you *pass* an item, without copying all of its attributes, to another G2 process simply by referring to its network handle.

If the system procedure fails, it signals an error. One possible error is cannot-register-non-item. Such an error could occur if you passed a text value to the system procedure, rather than an item. Other errors are: icp-interface-not-item, no-icp-interface-capability, no-icp-connection, and icp-connection-not-open.

If an item has not been network-registered, that is, the item does not have a handle, but is passed to or from a remote procedure declaring it as a handle, G2 registers the item automatically. You do not need to use the g2-register-on-network system procedure in such a case.

If the item that you pass to this system procedure is a gsi-data-service class (a GSI variable), and the ICP-interface that you are passing is a gsi-interface, the handle that the system procedure returns is the same as that used by the remote GSI process. Further, if the GSI variable that you register specifies the same gsi-

interface in its attribute table as you pass to the system procedure, GSI data service begins automatically.

Calling g2-register-on-network on an already registered item returns the existing handle.

One of the steps for item passing between G2 systems requires that you create a remote procedure declaration. When you create such a remote procedure declaration for item passing, you can specify a network handle. Use this system procedure in conjunction with the remote procedure item-passing syntax as handle, described in G2-to-G2 Interface in the *G2 Reference Manual*.

Example

This procedure iterates over every item upon a given workspace, and calls the system procedure to register each item.

```
network-registration(interface-workspace: class kb-workspace,
                                data-interface-object: class g2-to-g2-data-interface)
WS-ITEM: class item;
begin
    for WS-ITEM = each item upon interface-workspace
        do
            call g2-register-on-network(WS-ITEM, data-interface-object)
        end
end
```

Although you can register a **GSI variable** with more than one ICP-interface, the variable can only have one data server — the one in the gsi-interface-name attribute.

24

Priority Queue Operations

Describes the system procedure for manipulating priority queues.

Introduction

The priority-queue class provides a data structure for associating items with a priority, which can be any float value. For example, you could use a priority queue as the core of an event-based simulator, where the time an event should occur is used as the priority.

Use the priority queue operations for adding a new item to a queue at a given priority, removing an item from a queue, changing the priority of an existing item in a queue, getting the item with the highest priority, getting and removing the item with the highest priority, and determining if the queue is empty.

Note that if items in the priority queue have the same priority, the order in which they are retrieved from the queue is unpredictable. If you care about the order of items with the same priority, then provide a more detailed prioritization scheme, such as 1.1, 1.2, 1.3, and so on.

G2 provides priority queues as objects, which you can create from the KB Workspace > New Object menu or programmatically, using the create action.

Note G2 does not save the contents of a priority queue in a KB when it is saved. When a new KB is loaded, all priority queues are emptied. Also, disabling a priority queue empties its contents.

---

g2-change-priority-in-priority-queue

Changes the priority of an item in a priority queue.

Synopsis

g2-change-priority-in-priority-queue
(*queue*: class priority-queue, *item*: item, *new-priority*: float)
-> *success*: truth-value

Argument	Description
queue	The priority-queue that contains the specified *item*.
item	The item whose priority to change.
new-priority	The new priority, as a float.

Return Value	Description
success	True if the item exists in the queue, false otherwise.

Description

This procedure returns true if the item exists in the queue or false if it does not exist.

g2-clear-priority-queue

Removes all the items from a priority queue.

Synopsis

g2-clear-priority-queue
(*queue*: class priority-queue)

Argument	Description
queue	The priority-queue to clear.

Description

Removes all the items from the specified priority queue.

g2-get-highest-from-priority-queue

Returns the highest priority item from a priority queue.

Synopsis

g2-get-highest-from-priority-queue
(*queue*: class priority-queue)
-> *item*: item-or-value, *priority*: value

Argument	Description
queue	The priority-queue from which to get the highest priority item.

Return Value	Description
item	The item in the queue with the highest priority.
priority	The priority of the returned item.

Description

Returns the item in the queue with the highest priority, that is, with the smallest float value, and its associated priority. If the queue is empty, returns false, false. This procedure does not modify the queue.

g2-get-priority-in-queue

Gets the priority of a given item in a priority queue.

Synopsis

g2-get-priority-in-queue
(*queue*: class priority-queue, *item*: class item)
-> *success*: truth-value, *priority*: value

Argument	Description
queue	The priority-queue that contains the specified *item*.
item	The item whose priority to get.

Return Value	Description
success	True if the item is in the queue, false otherwise.
priority	The priority of the specified item or false if the item is not in the queue.

Description

This procedure returns false and false if the item is not in the queue.

g2-insert-in-priority-queue

Adds an item to a queue at a given priority.

Synopsis

g2-insert-in-priority-queue
(*queue*: class priority-queue, *item*: item, *priority*: float)
-> *success*: truth-value

Argument	Description
queue	The priority-queue in which to insert the specified *item*.
item	The item to insert.
priority	The priority of the inserted item, as a float.

Return Value	Description
success	True if the item was added, false if the item was already in the queue.

Description

This procedure only adds the item if it does not already exist in the queue.

g2-priority-queue-is-empty

Determines whether a priority queue is empty.

Synopsis

g2-priority-queue-is-empty
(*queue*: class priority-queue)
-> *success*: truth-value

Argument	Description
queue	The priority-queue to test.

Return Value	Description
success	True if the priority queue is empty, false otherwise.

Description

This procedure returns true if the queue is empty, false otherwise.

g2-remove-from-priority-queue

Removes an item from a priority queue.

Synopsis

g2-remove-from-priority-queue
(*queue*: class priority-queue, *item*: item)
-> *success*: truth-value

Argument	Description
queue	The priority-queue that contains the specified *item* to remove.
item	The item to remove.

Return Value	Description
success	True if the item was removed or false if the item does not exist in the queue.

Description

This procedure returns true if the item is removed or false if it does not exist in the queue.

g2-remove-highest-from-priority-queue

Returns the item in a priority queue with the highest priority

Synopsis

g2-remove-highest-from-priority-queue
(*queue*: class priority-queue)
-> *item*: item-or-value, *priority*: value

Argument	Description
queue	The priority-queue from which to remove the highest priority item.

Return Value	Description
item	The item in the queue with the highest priority.
priority	The priority of the returned item.

Description

This procedure returns the item with the highest priority from a priority queue, that is, with the smallest float value, and its associated priority, and removes the item from the queue. If the queue is empty, returns false, false.

25

Process Operations

Describes procedures that obtain information about the G2 process, that set a new password, that register a login handler, and that spawn, interrogate, and delete local and remote processes during KB execution.

Introduction

Use the process system procedures to launch processes, kill processes, and get G2 command-line arguments and environment variables.

g2-describe-g2-license

Returns the authorization information for the current G2 process.

Synopsis

g2-describe-g2-license
( )
-> *authorization*: text

Return Value	Description
authorization	A text string of the formatted description of the OK file used to authorized the current G2 process.

Description

This system procedure requires no arguments and returns a text value that includes a formatted description of the OK file used to authorize the current G2 process.

Example

A procedure that gets the current license information and displays it in a license-msg class of message is:

```
get-license-information(WS: class kb-workspace)
text-of-the-license: text;
M: class message;
begin
    text-of-the-license = call g2-describe-g2-license( );
    create a message M;
    transfer M to WS at (100, 100);
    change the text of M to text-of-the-license;
    make M permanent
end
```

g2-get-command-line-argument-from-launch

Returns a text string, which is the nth argument given to the command line that launched G2.

Synopsis

g2-get-command-line-argument-from-launch
(*index*: integer)
-> *argument*: text

Argument	Description
index	The index of the argument to return. The index is zero-based, with argument 0 being the name that invoked the G2 image. If the index is out of range, G2 signals an error.

Return Value	Description
argument	A text string of the command line option that launched G2, as specified by *index*.

Description

When you launch G2 from the command line, G2 interprets the command line as a string of 8-bit bytes representing Latin-1 characters. Any two-byte Unicode characters, such as Korean or Japanese characters, therefore appear as pairs of Latin-1 characters in an argument returned by this procedure.

Calling g2-get-command-line-argument-from-launch (0) returns different results on Windows and UNIX. On Windows, it always returns the fully qualified path name of the executable. On UNIX, it returns the exact path you used to launch G2. For example, if you `cd` to the *install-directory*`/g2` directory and enter `g2`, it returns simply `g2`; however, if you launch G2 from the top-level directory by entering *install-directory*`/g2/g2`, it returns the fully qualified path name to the executable.

You can provide user-defined command-line arguments when starting G2, using the `-init-string` command-line option, then use the g2-get-command-line-arguments-from-launch system procedure to access those user-defined arguments in G2. The information is stored in the window object inside G2. For example, you can use this argument when displaying Telewindows as an ActiveX control inside of a COM-compliant container. For an example, see the *Telewindows User's Guide*.

g2-get-g2-process-identifier

Returns the process ID of the current G2 process.

Synopsis

g2-get-g2-process-identifier
( )
-> *pid*: float

Return Value	Description
pid	The process ID of the G2 process.

Description

Returns the process ID of the current G2 process.

g2-kill-process

Kills a process running on the local system.

Synopsis

g2-kill-process
(*process-id*: float)
-> *process-killed*: truth-value

Argument	Description
process-id	The identification number of the process to be terminated.

Return Value	Description
process-killed	Returns true if the process was running and was successfully killed, and false otherwise.

Description

This procedure kills a process on the local system. It returns true if the process was running and was successfully killed, and false in any other case. Regardless of the outcome, the procedure reclaims any G2 storage associated with *process-id*.

Successfully killing a local process depends on the type of process spawned. For example, on Windows platforms, you cannot kill a spawned Windows process, but you can kill a command-line process. Similarly, you cannot kill all UNIX processes.

This procedure can kill only a process spawned by the G2 making the call. Processes spawned by other G2's, or by sources other than G2, are unaffected.

Note Use this procedure only to kill a local process. To kill a remote process, use g2-kill-remote-process.

Related Procedures

Use this procedure in conjunction with:

- g2-spawn-process-to-run-command-line
- g2-spawn-process-with-arguments
- g2-process-exists

Caution

When you spawn a process using g2-spawn-process-to-run-command-line or g2-spawn-process-with-arguments, G2 allocates memory to hold the process ID. To reclaim this memory, call g2-kill-process on the ID after the process completes. A KB that spawns an indefinitely large number of processes without reclaiming the storage holding their IDs will eventually consume all memory and halt G2.

g2-kill-remote-process

Kills a process running on a remote system.

Synopsis

g2-kill-remote-process
(*process-id*: float, *remote-win*: class ui-client-item, *timeout*: value)
-> *process-killed*: truth-value

Argument	Description
process-id	The identification number of the process to kill.
remote-win	The g2-window or ui-client-session item of the remote system.
timeout	The value of the timeout, which can be either a quantity, specifying the number of seconds that can pass before G2 halts procedure execution, or the value false, indicating that the system procedure should use the default timeout period. The default timeout is given by a `sys-mod.kb` integer parameter, default-remote-process-timeout, set to 30 seconds.

Return Value	Description
process-killed	Returns true if the process was running and was successfully killed.

Description

This procedure kills a process on a remote system connected to G2. It returns true if the process was running and was successfully killed, and signals an error in any other case. Regardless of the outcome, the procedure reclaims any G2 storage associated with *process-id*.

Successfully killing a remote process depends on the type of process spawned. For example, on Windows platforms, you cannot kill a spawned Windows process, but you can kill a command-line process. Similarly, you cannot kill all UNIX processes.

Related Procedures

Use this procedure in conjunction with:

- g2-spawn-remote-process-to-run-command-line
- g2-spawn-remote-process-with-arguments
- g2-remote-process-exists

Caution

- When you spawn a process, using g2-spawn-remote-process-to-run-command-line or g2-spawn-remote-process-with-arguments, G2 allocates memory to hold the process ID. To reclaim this memory, call g2-kill-remote-process on the ID after the process completes. A KB that spawns an indefinitely large number of processes without reclaiming the storage holding their IDs will eventually consume all memory and halt G2.
- G2 enters a wait state during execution of this procedure, which allows other processing that can asynchronously change the G2 context. Be sure to revalidate the context as needed after the procedure returns.

Example

The following procedure kills a remote process.

```
kill-remote-process(process-identification: float, window: class ui-client-item)
     = (truth-value)
process-killed: truth-value;
begin
     process-killed =
          g2-kill-remote-process(process-identification, window, false);
     return process-killed
end
```

g2-process-exists

Determines whether a given process exists on the local system.

Synopsis

g2-process-exists
(*process-id*: float)
-> *process-exists*: truth-value

Argument	Description
process-id	The identification number of the process.

Return Value	Description
process-exists	Truth-value that is true if the process exists, or false if it does not.

Description

G2-process-exists determines whether a process exists on the local system. It returns true if the process exists or false if it does not. The procedure signals an error if it is unable to determine whether the remote process exists.

This procedure can detect only a process spawned by the G2 making the call. Processes spawned by other G2's, or by sources other than G2, are inaccessible.

Note Use this procedure only to check on a local process. To check on a remote process, use g2-remote-process-exists.

Related Procedures

Use this procedure in conjunction with:

- g2-spawn-process-to-run-command-line
- g2-spawn-process-with-arguments
- g2-kill-process

g2-remote-process-exists

Determines whether a given process exists on a remote system.

Synopsis

g2-remote-process-exists
(*process-id*: float, *remote-win*: class ui-client-item, *timeout*: value)
-> *process-exists*: truth-value

Argument	Description
process-id	The identification number of the process.
remote-win	The g2-window or ui-client-session item of the remote system.
timeout	The value of the timeout, which can be either a quantity, specifying the number of seconds that can pass before G2 halts procedure execution, or the value false, indicating that the system procedure should use the default timeout period. The default timeout is given by a `sys-mod.kb` integer parameter, default-remote-process-timeout, set to 30 seconds. The system procedure signals an error in G2 if: • A timeout occurs. • The remote Telewindows or Telewindows2 Toolkit client does not support this system procedure. • The procedure is called with a local, rather than a remote, g2-window or ui-client-session item.

Return Value	Description
process-exists	Truth-value returns true if the process exists, or false if it does not.

Description

G2-remote-process-exists determines whether a given process is present on a remote system. It returns true if the process exists or false if it does not. This procedure signals an error if it is unable to determine whether the remote process exists.

This procedure can detect only a process spawned by the G2 making the call. Processes spawned by other G2's, or by sources other than G2, are inaccessible.

Note Use this procedure only to check on a remote process. To check on a local process, use g2-process-exists.

Related Procedures

Use this procedure in conjunction with:

- g2-spawn-remote-process-to-run-command-line
- g2-spawn-remote-process-with-arguments
- g2-kill-remote-process

Caution

G2 enters a wait state during execution of this procedure, which allows other processing that can asynchronously change the G2 context. Be sure to revalidate the context as needed after the procedure returns.

Example

This example returns a truth-value that indicates whether a given process ID number exists.

```
check-process-existence (process-identification: float,
                                 window: class ui-client-item)
    = (truth-value)
process-exists: truth-value;
begin
    process-exists =
        call g2-remote-process-exists(process-identification, window, false);
    return process-exists
end
```

g2-reroute-window

Reroutes the current g2-window or ui-client-session item from one G2 process to another.

Synopsis

g2-reroute-window
(*client*: class ui-client-item, *protocol*: text, *host*: text, *port*: text, *init-string*: text)

Argument	Description
client	The g2-window or ui-client-session item to reroute from one G2 process to another.
protocol	The network protocol currently in use by your system.
host	The network name of the system on which the second G2 process is running.
port	The port number of the new G2 process.
init-string	An initialization string of your choice.

Description

This system procedure lets you reroute a client from one G2 process to another (referred to here as the *source* G2 and the *target* G2, respectively). The procedure does not return a value.

When you call this procedure, G2 checks to see if the target G2, which is the name of the system in the *host* argument, is available. If it is, the g2-reroute-window procedure shuts down the source Telewindows or Telewindows2 Toolkit connection and reconnects the Telewindows or Telewindows2 Toolkit client to the target G2. The host names are looked up on the Telewindows (client) side, not on the G2 (server) side.

In this context, the term *available* means only that a G2 process is present on the target system. It does not imply any further capability, such as whether the KB is running or whether it includes corresponding support to reroute the client back to its source G2.

If the target G2 is not available, G2 neither provides an error message nor reroutes the *client*. To supply an appropriate message to the Telewindows or Telewindows2 Toolkit user, should a reroute failure occur, you need to provide your own error handling mechanism to inform the user that rerouting failed.

Hint G2 provides this system procedure as a building block with which to create Telewindows or Telewindows2 Toolkit rerouting capabilities for your KB. While the system procedure includes the basic rerouting functionality, you can add as few or as many features as your KB user requires.

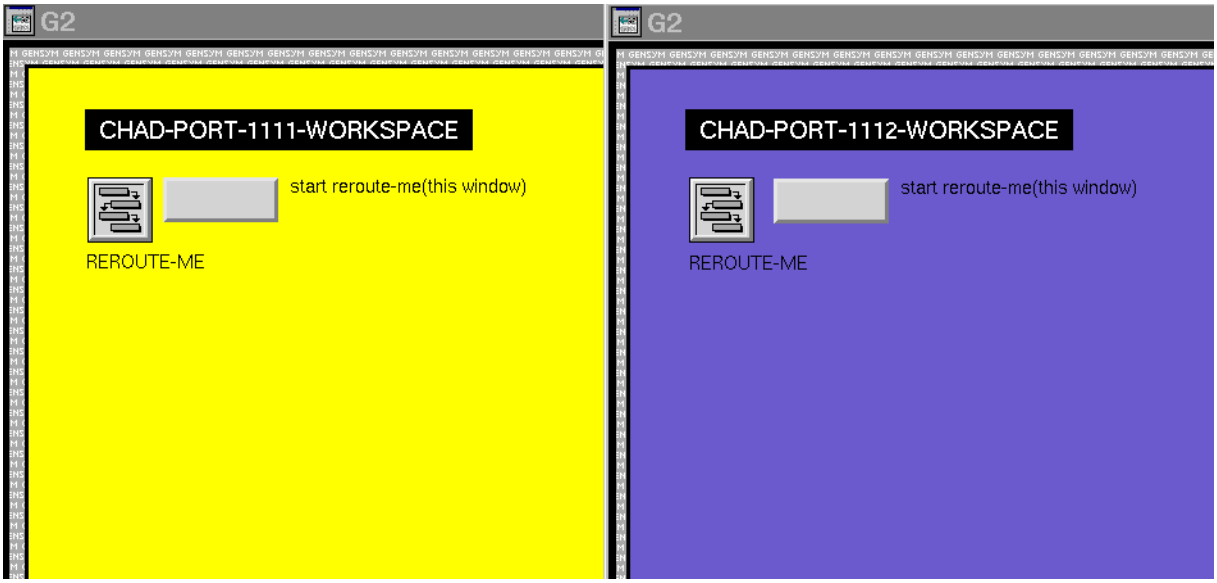
Example

One of the simplest ways to use the g2-reroute-window system procedure is to provide *corresponding functionality* in each KB that a Telewindows or Telewindows2 Toolkit user connects to. The example here uses this approach.

The purpose of the example that follows is twofold:

- To illustrate a minimal set of rerouting capabilities built around the system procedure
- To demonstrate the ease with which you can develop a custom Telewindows or Telewindows2 Toolkit rerouting module.

To begin the example, the next diagram shows the partial windows of two separate G2 processes, both running on host norwalk-n800c-2:



Each KB shows a single workspace whose name represents the host and port name (1111 or 1112) for the current G2 process. The KBs have corresponding functionality. Both include identical action buttons and similar procedures to handle the Telewindows client rerouting.

The procedures are shown next, calling out the different port numbers that each uses. Notice the last argument for the system procedures, which in both cases provides a descriptive text string.

NORWALK-N800C-2:1111

```
reroute-me(window: class ui-client-item)
{ procedure is on norwalk-n800c-2-port-1111-workspace }
begin
    call g2-reroute-window(window, "tcp-ip", "norwalk-n800c-2", "1112",
        "rerouting to port 1112")
end
```

NORWALK-N800C-2:1112

```
reroute-me(window: class ui-client-item)
{ procedure is on norwalk-n800c-2-port-1112-workspace }
begin
    call g2-reroute-window(window, "tcp-ip", "norwalk-n800c-2", "1111",
        "rerouting to port 1111")
end
```

If these KBs were running on two separate G2 processes, a Telewindows or Telewindows2 Toolkit client could initially connect to either. Once connected with the workspace visible, the Telewindows or Telewindows2 Toolkit user simply clicks on the action button that calls the reroute-me procedure to have G2 disconnect from the source process and reconnect to the target. In this case, since both G2 processes are running KBs with corresponding rerouting capabilities, the user can reroute *back* to the other process. The next section describes how to show the workspace to a Telewindows or Telewindows2 Toolkit user.

Additional Functionality

When a Telewindows or Telewindows2 Toolkit client connects to a KB, G2 creates a new g2-window or ui-client-session item representing that connection. By default, a Telewindows or Telewindows2 Toolkit user does not see a workspace unless the KB shows it.

One way to display a particular workspace to a Telewindows or Telewindows2 Toolkit user is to provide a whenever rule that checks for a new value in a particular g2-window attribute (g2-connection-status) or ui-client-session attribute (ui-client-session-status). The attributes automatically receives a value when G2 creates a g2-window or ui-client-session in response to a new Telewindows or Telewindows2 Toolkit connection.

A sample rule, showing what this example uses, follows:

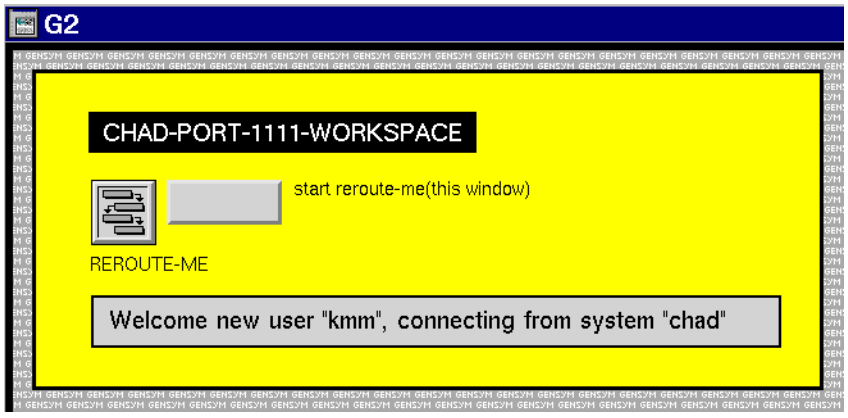
```
whenever the g2-connection-status of any g2-window G2W receives a value
    then start receive-tw(G2W, norwalk-n800c-2-1111-workspace)
```

Here is the procedure that the rule invokes:

```
receive-tw (window: class g2-window, ws: class kb-workspace)
M: class message;
begin
    show ws;
    create a message M;
    transfer M to ws at (-150, 150);
    change the text of M to "Welcome new user [the text of the
    g2-window-user-name-in-operating-system of window], connecting from system
    [the text of the g2-window-remote-host-name of window]";
    wait for 45 seconds;
    delete M
end
```

Each time the whenever rule detects a new value for the attribute, it passes that particular g2-window object (g2w, in this example), to the receive-tw procedure, whose text is also shown above. The procedure, in turn, also accepts the workspace as an argument, and then continues to post a welcome message containing information about the *rerouted* Telewindows or Telewindows2 Toolkit client.

Here is a sample of what the Telewindows user sees after rerouting to the target process at port 1111.



g2-spawn-process-to-run-command-line

Creates a subprocess on the local system to execute a command line.

Synopsis

g2-spawn-process-to-run-command-line
(*command*: text)
-> *process-id*: float

Argument	Description
command	The command line to be executed. Note that Windows 95 limits the length of a command line argument to 122 characters. Also, on Windows platforms, the *command* argument cannot refer to path names that include spaces.

Return Value	Description
process-id	A float value representing the identification number of the spawned process, which can be used by the g2-kill-process procedure. If a given process fails to launch, returns -1.

Description

The *command* argument allows full command-line interpreter syntax. That means, for example, that under UNIX and Windows one can use pipes (|), variable substitution, I/O redirection (<, >), etc. Under UNIX, Bourne shell syntax is required.

Under both UNIX and Windows, a command shell process is started that interprets the command. Thus, the spawned-process-id that is returned is that of the command shell rather than that of the process that the command shell spawns.

To refer to a pathname with spaces on Windows platforms, you must surround the pathname with quotes, for example:

```
call g2-spawn-process-to-run-command-line
  ("@"[command-file-name]@" [command-argument]");
```

Related Procedures

Use this procedure in conjunction with:

- g2-spawn-remote-process-with-arguments
- g2-process-exists
- g2-kill-process

Caution

When you spawn a process, G2 allocates memory to hold the process ID. To reclaim this memory, call g2-kill-process on the ID after the process completes. A KB that spawns an indefinitely large number of processes without reclaiming the storage holding their IDs will eventually consume all memory and halt G2.

Example

You may want to issue commands, similar to operating-system commands, from within G2. For example, the following procedure lists all of the processes running on your machine and how much memory is available: HQ-5576355 Doc: g2-spawn-process-to-run-command-line typos

```
spawn-process(ws: class kb-workspace)
process-status: float;
this-stream: class g2-stream;
this-line: text;
begin
    call g2-change-default-directory("/disc/examples/new");
    process-status =
        call g2-spawn-process-to-run-command-line("ps -aux > process-file");
    if process-status /= -1
        then post for the next 30 seconds "Process has been spawned.";
    repeat
        wait for 1 second;
        process-exists = call g2-process-exists(process-status);
    exit if process-exists = false;
    end;
    this-stream = call g2-open-file-for-read("/disc/examples/new/process-file");
    transfer this-stream to ws;
    repeat
        if the g2-stream-status of this-stream is end-of-file-reached
            then post for the next 30 seconds
                "[the g2-stream-status of this-stream]";
        exit if the g2-stream-status of this-stream is end-of-file-reached;
        this-line = call read-line(this-stream);
        post for the next 30 seconds "[this-line]"
    end;
    call g2-close-file(this-stream);
end
```

g2-spawn-process-with-arguments

Creates a subprocess on the local system to execute a command directly.

Synopsis

g2-spawn-process-with-arguments
(*command*: text)
-> *process-id*: float

Argument	Description
command	The command and its arguments to be spawned. Note that Windows 95 limits the length of a command line argument to 122 characters.

Return Value	Description
process-id	A float value representing the identification number of the spawned process, which can be used by the g2-kill-process procedure. If a given process fails to launch, returns -1.

Description

This is invoked with a string that contains a command and an optional list of arguments. A process is spawned to execute the command with the given arguments.

This procedure differs from g2-spawn-process-to-run-command-line because no intermediate command shell is spawned. The returned spawned-process-id is that of the command. Arguments are passed directly to the spawned command with no special treatment except for that described below. This means that under UNIX and Windows, no command-shell features are allowed. Thus, pipes (|), variable substitution, I/O redirection (<, >), etc., are disallowed.

Under UNIX, any characters in the command text that are surrounded by a pair of single quotes (') or double quotes (") are passed as a single argument. This allows spaces or quote marks to be passed to the command within arguments. For example, to pass a double quote, surround the argument containing it with single quotes, or vice versa.

Under Windows, the command and arguments are passed directly as a single string. The argument list is not preprocessed in any way.

To refer to path names with spaces, you must surround the path name with an escaped set of quotes, for example:

```
start g2-spawn-process-with-arguments ("@"C:\Program Files\Gensym\
nol-4.3r1\nolstudio\nolonline.bat@" -g2host GCHEN-N800C -g2port 1111
-interfaceclass nols-gateway -interfacename NOLS-INTERFACE
-listenerport 22044 ")
```

Related Procedures

Use this procedure in conjunction with:

- g2-spawn-process-to-run-command-line
- g2-process-exists
- g2-kill-process

Caution

When you spawn a process, G2 allocates memory to hold the process ID. To reclaim this memory, call g2-kill-process on the ID after the process completes. A KB that spawns an indefinitely large number of processes without reclaiming the storage holding their IDs will eventually consume all memory and halt G2.

g2-spawn-remote-process

This procedure has been renamed g2-spawn-remote-process-to-run-command-line. Although the superseded name will continue to be supported, the new name should be used in all new code.

Caution

G2 enters a wait state during execution of this procedure, which allows other processing that can asynchronously change the G2 context. Be sure to revalidate the context as needed after the procedure returns.

When you spawn a process, G2 allocates memory to hold the process ID. To reclaim this memory, call g2-kill-remote-process on the ID after the process completes. A KB that spawns an indefinitely large number of processes without reclaiming the storage holding their IDs will eventually consume all memory and halt G2.

g2-spawn-remote-process-to-run-command-line

Creates a subprocess on a remote system to execute a command line.

Note The procedure replaces g2-spawn-remote-process, which will continue to be supported but should not be used in new code.

Synopsis

g2-spawn-remote-process-to-run-command-line
(*command*: text, *remote-win*: class ui-client-item, *timeout*: value)
-> *process-id*: float

Argument	Description
command	The command line to be executed on the remote system.
remote-win	The g2-window or ui-client-session item of the remote system.
timeout	The value of the timeout, which can be either a quantity, specifying the number of seconds that can pass before G2 halts procedure execution, or the value false, indicating that the system procedure should use the default timeout period. The default timeout is given by a `sys-mod.kb` integer parameter, default-remote-process-timeout, set to 30 seconds. The system procedure signals an error in G2 if: • A timeout occurs. • The remote process does not support this system procedure. • The procedure is called with a local, rather than a remote, g2-window or ui-client-session item.

Value	Description
process-id	The identification number of the spawned process on the remote system as a float.

Description

G2-spawn-remote-process-to-run-command-line permits a G2 server to spawn a process on a Telewindows or Telewindows2 Toolkit client, returning the remote process ID number if the spawn is successful, or signalling an error if it is not.

This procedure is identical to g2-spawn-process-to-run-command-line insofar as it allows full command-line syntax and spawns an intermediate command-line interpreter.

Related Procedures

Use this procedure in conjunction with:

- g2-spawn-remote-process-with-arguments
- g2-remote-process-exists
- g2-kill-remote-process

Caution

- When you spawn a process, G2 allocates memory to hold the process ID. To reclaim this memory, call g2-kill-remote-process on the ID after the process completes. A KB that spawns an indefinitely large number of processes without reclaiming the storage holding their IDs will eventually consume all memory and halt G2.
- G2 enters a wait state during execution of this procedure, which allows other processing that can asynchronously change the G2 context. Be sure to revalidate the context as needed after the procedure returns.
- The *command* argument to g2-spawn-process-to-run-command-line cannot refer to path names that include spaces. Therefore, on Windows platforms, to refer to a path name with spaces, you must surround the path name with quotes, for example:

```
call g2-spawn-process-to-run-command-line
  ("@"[command-file-name]@" [command-argument]");
```

Example

This example of g2-spawn-remote-process-to-run-command-line invokes a browser utility, and returns the process ID:

```
start-remote-browser( ) = (float)
process-identification: float;
begin
    process-identification =
        call g2-spawn-remote-process-to-run-command-line("/home/abc/net",
            process-window, false);
    return process-identification
end
```

g2-spawn-remote-process-with-arguments

Creates a subprocess on a remote system to execute a command directly.

Synopsis

g2-spawn-remote-process-with-arguments
(*command*: text, *remote-win*: class ui-client-item, *timeout*: value)
-> *process-id*: float

Argument	Description
command	The command and its arguments to be spawned on the remote system.
remote-win	The g2-window or ui-client-session item of the remote system.
timeout	The value of the timeout, which can be either a quantity, specifying the number of seconds that can pass before G2 halts procedure execution, or the value false, indicating that the system procedure should use the default timeout period. The default timeout is given by a `sys-mod.kb` integer parameter, default-remote-process-timeout, set to 30 seconds. The system procedure signals an error in G2 if: • A timeout occurs. • The remote process does not support this system procedure. • The procedure is called with a local, rather than a remote, g2-window or ui-client-session item.

Value	Description
process-id	The identification number of the spawned process on the remote system as a float.

Description

G2-spawn-remote-process-with-arguments permits a G2 server to spawn a process on a Telewindows or Telewindows2 Toolkit client, returning the remote process ID number if the spawn is successful, or signalling an error if it is not.

This procedure is identical to g2-spawn-process-with-arguments insofar as it spawns a command with the given arguments. No intermediate command-line interpreter is spawned.

Related Procedures

Use this procedure in conjunction with:

- g2-spawn-remote-process-to-run-command-line
- g2-remote-process-exists
- g2-kill-remote-process

Caution

- When you spawn a process, G2 allocates memory to hold the process ID. To reclaim this memory, call g2-kill-remote-process on the ID after the process completes. A KB that spawns an indefinitely large number of processes without reclaiming the storage holding their IDs will eventually consume all memory and halt G2.
- G2 enters a wait state during execution of this procedure, which allows other processing that can asynchronously change the G2 context. Be sure to revalidate the context as needed after the procedure returns.

26

Profiling Operations

Describes procedures that control profiling during KB execution.

Introduction

Use the profiling system procedures to enable, disable, and use system profiling in a knowledge base (KB). Profiling a KB creates an instance of a system-profile-information item, containing the profile information gathered about the profiled KB.

For a complete description of profiling, and the types of activities G2 can profile, see Profiling and KB Performance in the *G2 Reference Manual*.

These procedures also allow you to get performance statistics.

g2-clear-profile

Clears all profile data.

Synopsis

g2-clear-profile
( )

Description

This system procedure clears all of the profiled data collected since the g2-enable-profiling system procedure was called. This procedure does not require any arguments, nor does it return a value.

g2-disable-profiling

Disables profiling.

Synopsis

g2-disable-profiling
( )

Description

This system procedure disables profiling. It does not require any arguments, nor does it return a value. Calling g2-disable-profiling stops profiling collection, but does not clear the profiling data collected since profiling was last enabled.

g2-enable-profiling

Enables system profiling.

Synopsis

g2-enable-profiling
( )

Description

This system procedure enables system profiling in a KB. It does not require any arguments, nor does it return a value.

Profiling begins as soon as the system procedure returns. For instance, if the g2-enable-profiling procedure is called within a procedure, G2 profiles any statements following that statement.

Note Profiling does not track activities performed by the same item from which profiling is enabled.

Example

The following procedure illustrates how to call each of the profiling system procedures:

```
profiling-test(ws: kb-workspace)
profile: class system-profile-information;
begin
    call g2-enable-profiling( );
    profile = call g2-get-profiled-information( );
    transfer profile to ws;
    call g2-clear-profile( );
    call g2-disable-profiling( )
end
```

g2-get-profiled-information

Gets all profiled data.

Synopsis

g2-get-profiled-information
( )
-> *profiled-data*: class system-profile-information

Return Value	Description
profiled-data	A system-profile-information item with all data profiled since the g2-enable-profiling system procedure was placed and returned.

Description

This system procedure does not require any arguments, but returns an instance of a system-profile-information object. The object instance contains all profiling information collected since profiling was first enabled (or cleared).

g2-name-for-item

Returns the name of any item.

Synopsis

g2-name-for-item
(*item*: class item)
-> *item-name*: value

Argument	Description
item	Any KB item whose user- or system-provided name you wish to obtain.

Return Value	Description
item-name	A value indicating the name of *item*.

Description

This system procedure returns the name of any item, whether it is a user-provided name as a symbol or the system-generated internal name as a text value. For an item that has more than one user-provided name, the procedure returns a single, unique name.

While G2 profiling requires the use of this system procedure, its use is not limited to system profiling.

Example

This procedure illustrates how to get the names of all items on a given workspace, and display the results to the Message Board:

```
get-names(ws: class kb-workspace)
I: class item;
N: value;
T: text;
begin
     T = "These are the names of all items on [the name of ws]:";
     for I = each item upon ws
          do
               N = call g2-name-for-item(I);
               T = "[T] [N],";
          end;
     post "[T]"
end
```

g2-get-performance-frequency

Returns the number of "ticks" per second on the machine.

Synopsis

g2-get-performance-frequency
( )
-> *ticks*: integer

Return Value	Description
ticks	The number of ticks per second in G2, as an integer.

Description

On Windows, this system procedure returns a highly precise timestamp, which uses the Microsoft high-resolution timer.

This number is useful only for interpreting the results of the g2-get-performance-counter and g2-get-and-log-performance-counter system procedures.

g2-get-performance-counter

Returns the current number of "ticks" as a float and logs the output.

Synopsis

g2-get-performance-counter
( )
-> *ticks*: float

Return Value	Description
ticks	The current number of ticks in G2, as a float.

Description

Use this procedure to get performance statistics for executing a task in G2.

On Windows, this system procedure returns a highly precise timestamp, which uses the Microsoft high-resolution timer.

All output goes to the standard G2 output. On Windows, standard output is usually a log file, but it could be a console window if G2 is run with the -no-log command-line option. On UNIX, standard output is usually the shell where G2 was started, unless G2's output is redirected into a file. If tracing-and-breakpoints is enabled, the output is also logged to that file.

g2-get-and-log-performance-counter

Returns the current number of "ticks" as a float and logs the output.

Synopsis

g2-get-and-log-performance-counter
(*msg*: text)
-> *ticks*: float

Return Value	Description
ticks	The current number of ticks, as a float.s

Description

Use this procedure to get performance statistics for executing a task and log that information.

On Windows, this system procedure returns a highly precise timestamp, which uses the Microsoft high-resolution timer.

The message is formatted on two lines, as follows:

High Precision Counter: *counter-as-64-bit-int*
msg

All output goes to the standard G2 output. On Windows, standard output is usually a log file, but it could be a console window if G2 is run with the -no-log command-line option. On UNIX, standard output is usually the shell where G2 was started, unless G2's output is redirected into a file. If tracing-and-breakpoints is enabled, the output is also logged to that file.

27

Publish/Subscribe Operations

Describes procedures that allow you to subscribe to G2 item events.

Introduction

G2 provides a publish/subscribe facility, which allows application developers to implement scalable, distributed applications that can respond dynamically to changes in the application. This facility provides tools similar to what many modern development environments such as Java provide.

The publish/subscribe facility allows applications to subscribe to these item events:

- modify — When the value of a single attribute, a list of attributes, or all attributes of an item change. This event also occurs when the value of a variable or parameter changes.
- item-color-pattern-change — When any color region of an icon changes.
- delete — When an item is deleted.
- add-item-to-workspace — When an item is added to a workspace.
- remove-item-from-workspace — When an item is removed from a workspace.
- custom-event — When a custom event is sent.

The publish/subscribe facility also provides a way of sending custom events.

When the event occurs, G2 can execute a callback procedure, which can refer to user-defined data passed to the procedure that creates the subscription. The subscription procedure returns handles, which you must use to remove subscription registrations.

Note that G2 schedules callbacks to occur in response to an event; the event does not automatically cause the callback to execute, then wait for it to finish.

When an item is deleted, all subscriptions on it are automatically deregistered.

The publish/subscribe facility supports remote application subscriptions and callbacks.

Note Subscriptions are not saved in the KB; however, they remain after resetting G2.

Callback Signatures

The callback signatures depend on the type of event.

You will receive a level 3 warning at subscription time and an error at invocation time if you subscribe to a local callback that uses the wrong signature.

General Callback

The signature of the callback procedure for all subscription procedures except for g2-subscribe-to-add-item-to-workspace, g2-subscribe-to-remove-item-from-workspace, and g2-subscribe-to-variable-or-parameter-value:

```
my-callback
    (event: symbol, item: class item | integer,
    denotation-sequence: sequence, new-value: item-or-value,
    user-data: item-or-value, subscription-handle: integer)
```

Argument	Description
event	A symbol representing the type of event notification. The options are: modify, item-color-pattern-change, delete, and custom-event.
item	When invoked locally, the subscription item. When invoked remotely, the network handle of the item that is registered remotely, which is an integer. For more information, see Registering Callbacks Remotely.
denotation-sequence	A sequence of structures that provides information about the event. See below for examples for each type of event.
new-value	The new value of the changed attribute. When the *denotation-sequence* is for a color change, the *new-value* is a structure, which is a subset of the item-color-pattern hidden attribute, depending on which color regions have changed. See g2-subscribe-to-item-color-pattern-change.
user-data	The *user-data* specified in the subscription procedure.
subscription-handle	The subscription handle returned by the subscription procedure.

If the attribute named temp has changed, the denotation sequence looks like this:

```
sequence (structure
(TYPE: the symbol ATTRIBUTE,
NAME: the symbol TEMP))
```

If a color region of the item changes, the sequence looks like this:

```
sequence (structure
(TYPE: the symbol ITEM-COLOR-PATTERN-CHANGE))
```

See g2-subscribe-to-item-color-pattern-change.

If the item is deleted, the sequence looks like this:

```
sequence (structure
(TYPE: the symbol ATTRIBUTE,
NAME: the symbol NAMES))
```

If the item receives a custom event named my-custom-event, the sequence looks like this:

```
sequence (structure
(TYPE: the symbol CUSTOM-EVENT,
CUSTOM-EVENT-NAME: the symbol MY-CUSTOM-EVENT))
```

See g2-subscribe-to-item-deletion.

Callback for Workspace Events

The signature of the callback procedure for g2-subscribe-to-add-item-to-workspace and g2-subscribe-to-remove-item-from-workspace is:

```
my-callback
    (event: symbol, workspace: class kb-workspace | integer,
    denotation-sequence: sequence, item: class item | integer,
    user-data: item-or-value, subscription-handle: integer)
```

Argument	Description
event	A symbol representing the type of event notification. The options are: add-item-to-workspace and remove-item-from-workspace,
workspace	When invoked locally, the subscription workspace. When invoked remotely, the network handle of the workspace that is registered remotely, which is an integer. For more information, see Registering Callbacks Remotely.
denotation-sequence	A sequence of structures that provides information about the event. See below for examples of each type of event.

Argument	Description
item	The item added to or removed from the workspace.
user-data	The *user-data* specified in the subscription procedure.
subscription-handle	The subscription handle returned by the subscription procedure.

Note When an item is removed from a workspace, the callback is invoked *after* the item has already been removed. Therefore, the body of the callback needs to perform an existence check when referring to the *item* argument. If the item has been deleted from the KB, the callback is invoked with no value for the *item* argument. If, however, it has been removed from the workspace but not deleted from the KB, the callback can still refer to the *item* argument. For an example, see Publish/Subscribe Facility in the *G2 Reference Manual*.

If an item is added to a workspace, the sequence looks like this:

```
sequence (structure
(TYPE: the symbol ADD-ITEM-TO-WORKSPACE)
```

If an item is removed from a workspace, the sequence looks like this:

```
sequence (structure
(TYPE: the symbol REMOVE-ITEM-FROM-WORKSPACE)
```

Callback for Variable or Parameter Events

The signature of the callback procedure for g2-subscribe-to-variable-or-parameter-value is:

```
my-callback
    (event: symbol, item: class item | integer,
    denotation-sequence: sequence, new-value: item-or-value,
    user-data: item-or-value, subscription-handle: integer)
```

Argument	Description
event	The symbol modify.
item	When invoked locally, the subscription item. When invoked remotely, the network handle of the item that is registered remotely, which is an integer. For more information, see Registering Callbacks Remotely.
denotation-sequence	A sequence of structures that provides information about the event. See below for an example.
new-value	The new value of the variable or parameter.
user-data	The *user-data* specified in the subscription procedure.
subscription-handle	The subscription handle returned by the subscription procedure.

The denotation sequence looks like this for both variables and parameters:

```
sequence (structure
(TYPE: the symbol VARIABLE-VALUE))
```

Registering Callbacks Remotely

To register callbacks remotely, you need to make an RPC call into G2 to a procedure that does the registration. When registering callbacks remotely, you pass a symbol as the *callback* argument, which is interpreted as the name of a procedure in the remote registration. If no such procedure exists, an error is thrown at when the callback is invoked.

Before invoking the callback remotely, any items being passed to the callback are registered for network use as if the g2-register-on-network system procedure had been called on them and their associated interface. Registering the item returns a network handle, as an integer, which is passed as the *item* argument to the callback procedure.

Note When invoking callbacks in a bridge, you must retrieve the network handle by calling `gsi_handle_of`, not `gsi_int_of`. For more information, see the *G2 Gateway User's Guide*.

For examples, see Publish/Subscribe Facility in the *G2 Reference Manual*.

g2-subscribe-to-item-attributes

Invokes subscribed callbacks when attributes of an item change.

Synopsis

g2-subscribe-to-item-attributes
(*item* class: item, *attribute-spec*: value, *callback*: symbol | class procedure, *user-data*: item-or-value)
-> *return-value*: integer | sequence

Argument	Description
item	The item to which to subscribe.
attribute-spec	The symbol all, which means subscribe to all attributes, a symbol, which is the attribute name to which to subscribe, or a sequence of symbols, which is a list of attribute names.
callback	When called locally, a procedure to invoke when the specified attribute is modified, or a symbol that is the name of a procedure. When specifying a symbol, the procedure must exist at subscription time; otherwise, an error is signalled. You will receive a level 3 warning at subscription time and an error at invocation time if you subscribe to a local callback that uses the wrong signature. When called remotely, a procedure to invoke across the network or a symbol that is the name of a procedure. In the case of remote callbacks, the procedure must exist at invocation time; otherwise, an error is signalled. The callback argument must be a procedure or a symbol; otherwise, an error is signalled. For more information, see Registering Callbacks Remotely.
user-data	An item or value passed to the callback when it is invoked. For example, you can use the same callback for multiple registrations and specify different *user-data* to differentiate each callback.

Return Value	Description
return-value	When *attribute-spec* is an attribute or the symbol all, returns an integer that represents the registration. When *attribute-spec* is a sequence of attributes, returns a sequence of integers that represents the registration of each attribute.

Description

The subscription can be to a single attribute, a sequence of attributes, or all attributes.

The callback should check the event argument to determine what type of event has been triggered so it only triggers when the event type is modify.

For the syntax of the callback, see Callback Signatures.

For an example, see Publish/Subscribe Facility in the *G2 Reference Manual*.

g2-subscribe-to-item-color-pattern-change

Invokes subscribed callbacks when any region of an item's icon changes.

Synopsis

g2-subscribe-to-item-color-pattern-change
(*item*: class item, *callback*: symbol | class procedure, *user-data*: item-or-value)
-> *handle:* integer

Argument	Description
item	The item to which to subscribe.
callback	When called locally, a procedure to invoke when the icon color changes, or a symbol that is the name of a procedure. When specifying a symbol, the procedure must exist at subscription time; otherwise, an error is signalled. You will receive a level 3 warning at subscription time and an error at invocation time if you subscribe to a local callback that uses the wrong signature. When called remotely, a procedure to invoke across the network or a symbol that is the name of a procedure. In the case of remote callbacks, the procedure must exist at invocation time; otherwise, an error is signalled. The callback argument must be a procedure or a symbol; otherwise, an error is signalled. For more information, see Registering Callbacks Remotely.
user-data	An item or value passed to the callback when it is invoked. For example, you can use the same callback for multiple registrations and specify different *user-data* to differentiate each callback.

Return Value	Description
subscription-handle	An integer that represents the registration.

Description

The callback should check the event argument to determine what type of event has been triggered so it only triggers when the event type is item-color-pattern-change.

For the syntax of the callback, see Callback Signatures.

g2-subscribe-to-item-deletion

Invokes subscribed callbacks when an item is deleted.

Synopsis

g2-subscribe-to-item-deletion
(*item*: class item, *callback*: symbol | class procedure, *user-data*: item-or-value)
-> *handle*: integer

Argument	Description
item	The item to which to subscribe.
callback	When called locally, a procedure to invoke when the specified item is deleted, or a symbol that is the name of a procedure. When specifying a symbol, the procedure must exist at subscription time; otherwise, an error is signalled. You will receive a level 3 warning at subscription time and an error at invocation time if you subscribe to a local callback that uses the wrong signature. When called remotely, a procedure to invoke across the network or a symbol that is the name of a procedure. In the case of remote callbacks, the procedure must exist at invocation time; otherwise, an error is signalled. The callback argument must be a procedure or a symbol; otherwise, an error is signalled. For more information, see Registering Callbacks Remotely.
user-data	An item or value passed to the callback when it is invoked. For example, you can use the same callback for multiple registrations and specify different *user-data* to differentiate each callback.

Return Value	Description
handle	An integer that represents the registration.

Description

The callback should check the event argument to determine what type of event has been triggered so it only triggers when the event type is delete.

Note that by the time the callback is run, the registered item has already been deleted, so if you try to access the *item* argument to the callback, the callback will signal an error. One technique you can use when registering the callback is to pass the UUID or other identifying information as the *user-data* argument to identify the deleted item.

For the syntax of the callback, see Callback Signatures.

For an example, see Publish/Subscribe Facility in the *G2 Reference Manual*.

g2-subscribe-to-add-item-to-workspace

Invokes subscribed callbacks when an item is added to a workspace.

Synopsis

g2-subscribe-to-add-item-to-workspace
(*workspace:* class kb-workspace, *callback*: symbol | class procedure,
user-data: item-or-value)
-> *handle:* integer

Argument	Description
workspace	The workspace to which to subscribe.
callback	When called locally, a procedure to invoke when an item is added to the specified workspace, or a symbol that is the name of a procedure. When specifying a symbol, the procedure must exist at subscription time; otherwise, an error is signalled. You will receive a level 3 warning at subscription time and an error at invocation time if you subscribe to a local callback that uses the wrong signature. When called remotely, a procedure to invoke across the network or a symbol that is the name of a procedure. In the case of remote callbacks, the procedure must exist at invocation time; otherwise, an error is signalled. The callback argument must be a procedure or a symbol; otherwise, an error is signalled. For more information, see Registering Callbacks Remotely.
user-data	An item or value passed to the callback when it is invoked. For example, you can use the same callback for multiple registrations and specify different *user-data* to differentiate each callback.

Return Value	Description
handle	An integer that represents the registration.

Description

The callback is invoked for each item added to the workspace.

The callback should check the event argument to determine what type of event has been triggered so it only triggers when the event type is add-item-to-workspace.

The callback for this procedure is slightly different from the callback for the other subscription procedures. For the syntax of the callback, see Callback Signatures.

g2-subscribe-to-remove-item-from-workspace

Invokes subscribed callbacks when an item is removed from a workspace.

Synopsis

g2-subscribe-to-remove-item-from-workspace
(*workspace:* class kb-workspace, *callback*: symbol | class procedure,
user-data: item-or-value)
-> *handle:* integer

Argument	Description
workspace	The workspace to which to subscribe.
callback	When called locally, a procedure to invoke when an item is removed from the specified workspace, or a symbol that is the name of a procedure. When specifying a symbol, the procedure must exist at subscription time; otherwise, an error is signalled. You will receive a level 3 warning at subscription time and an error at invocation time if you subscribe to a local callback that uses the wrong signature. When called remotely, a procedure to invoke across the network or a symbol that is the name of a procedure. In the case of remote callbacks, the procedure must exist at invocation time; otherwise, an error is signalled. The callback argument must be a procedure or a symbol; otherwise, an error is signalled. For more information, see Registering Callbacks Remotely.
user-data	An item or value passed to the callback when it is invoked. For example, you can use the same callback for multiple registrations and specify different *user-data* to differentiate each callback.

Return Value	Description
handle	An integer that represents the registration.

Description

The callback is invoked for each item removed from the workspace.

The callback should check the event argument to determine what type of event has been triggered so it only triggers when the event type is remove-item-from-workspace.

The callback is invoked *after* the item has been removed from the workspace; therefore, the body of the callback should check whether the item argument to the callback exists to avoid an error. For more information and for the syntax for the callback, see Callback Signatures.

g2-subscribe-to-variable-or-parameter-value

Invokes subscribed callbacks when the value of a variable or parameter changes from its previous value, or when the value of a variable expires and the variable receives a new value.

Synopsis

g2-subscribe-to-variable-or-parameter-value
(*var-or-param*: variable-or-parameter, *callback*: symbol | class procedure, *user-data*: item-or-value)
-> *handle*: integer

Argument	Description
var-or-param	The variable or parameter to which to subscribe.
callback	When called locally, a procedure to invoke when the specified variable or parameter value changes, or a symbol that is the name of a procedure. When specifying a symbol, the procedure must exist at subscription time; otherwise, an error is signalled. You will receive a level 3 warning at subscription time and an error at invocation time if you subscribe to a local callback that uses the wrong signature. When called remotely, a procedure to invoke across the network or a symbol that is the name of a procedure. In the case of remote callbacks, the procedure must exist at invocation time; otherwise, an error is signalled. The callback argument must be a procedure or a symbol; otherwise, an error is signalled. For more information, see Registering Callbacks Remotely.
user-data	An item or value passed to the callback when it is invoked. For example, you can use the same callback for multiple registrations and specify different *user-data* to differentiate each callback.

Return Value	Description
handle	An integer that represents the registration.

Description

The callback should check the event argument to determine what type of event has been triggered so it only triggers when the event type is modify.

For the syntax of the callback, see Callback Signatures.

For an example, see Publish/Subscribe Facility in the *G2 Reference Manual*.

g2-subscribe-to-custom-event

Invokes subscribed callbacks when a user-defined event occurs.

Synopsis

g2-subscribe-to-custom-event
(*item*: class item, *custom-event-name*: symbol,
callback: symbol | class procedure, *user-data*: item-or-value)
-> *handle*: integer

Argument	Description
item	The item to which to subscribe.
custom-event-name	A symbol naming a user-defined event.
callback	When called locally, a procedure to invoke when the specified custom event occurs, or a symbol that is the name of a procedure. When specifying a symbol, the procedure must exist at subscription time; otherwise, an error is signalled. You will receive a level 3 warning at subscription time and an error at invocation time if you subscribe to a local callback that uses the wrong signature. When called remotely, a procedure to invoke across the network or a symbol that is the name of a procedure. In the case of remote callbacks, the procedure must exist at invocation time; otherwise, an error is signalled. The callback argument must be a procedure or a symbol; otherwise, an error is signalled. For more information, see Registering Callbacks Remotely.
user-data	An item or value passed to the callback when it is invoked. For example, you can use the same callback for multiple registrations and specify different *user-data* to differentiate each callback.

Return Value	Description
handle	An integer that represents the registration.

Description

You send user-defined events by using g2-send-notification-to-item.

For the syntax of the callback, see Callback Signatures.

g2-send-notification-to-item

Sends a custom event to an item.

Synopsis

g2-send-notification-to-item
(*item*: class item, *custom-event-name*: symbol, *new-value*: item-or-value)

Argument	Description
item	The item to notify.
custom-event-name	A symbol naming a user-defined event.
new-value	The new value for the event.

Description

You can specify a new value for the event, which is passed to the *new-value* argument of the callback. You subscribe to user-defined events by using g2-subscribe-to-custom-event.

For the syntax of the callback, see Callback Signatures.

g2-deregister-subscription

Removes a subscription registration.

Synopsis

g2-deregister-subscription
(*handle*: integer)

Argument	Description
handle	An integer handle that represents a subscription.

Description

Removes a subscription registration given a handle, which is returned from a subscription registration.

Note Canceling a subscription by calling this procedure also cancels all active subscriptions to the event in that G2 from all clients, such as G2 ActiveXLink and G2 JavaLink; however, the client is not notified of the cancellation. If you then try to unsubscribe in the client, G2 reports an error and the client throws an exception. You should not subscribe to events in the client and unsubscribe from the event in G2.

g2-get-subscription-handle-info

Returns a structure that describes a subscription handle.

Synopsis

g2-get-subscription-handle-info
(*handle*: integer)
-> *info*: structure

Argument	Description
handle	An integer handle that represents a subscription.

Return Value	Description
info	A structure that describes the subscription. See below for the syntax.

Description

Returns a structure with the following syntax:

structure
(SUBSCRIBED-ITEM: *item*,
SUBSCRIBED-CALLBACK: *procedure-or-symbol*,
SUBSCRIBED-DENOTATION: *sequence*,
SUBSCRIPTION-USER-DATA: *item-or-value*)

where:

- subscribed-item is the item that is being subscribed to.
- subscribed-callback is the local callback procedure or a symbol naming the remote callback.
- subscription-denotation is a sequence of the form passed to the callbacks. In the case of an attribute-modification subscription, it names the attribute associated with the subscription, or ALL to indicate a subscription to all attributes of the item.
- subscription-user-data is the *user-data* specified with the subscription.

For example:

sequence (structure
(TYPE: the symbol attribute,
NAME: the symbol ALL))

If you have subscribed to item-deletion events, the denotation-sequence has this format:

```
sequence (structure
(TYPE: the symbol ATTRIBUTE,
NAME: the symbol NAMES))
```

The procedure generates an error if passed an invalid handle.

g2-get-subscription-handles-for-items

Returns a sequence of all subscription handles for the particular item.

Synopsis

g2-get-subscription-handles-for-item
(*item*: class item)
-> *handles*: sequence

Argument	Description
item	The item whose subscription handles to get.

Argument	Description
handles	A sequence of integer handles that represents the subscriptions for the item.

Description

If the item has no subscriptions, this procedure returns an empty sequence.

28

Rule Operations

Describes procedures that get information about forward and backward chaining to rules.

Introduction

Use the rule operations system procedures to get information on rules that cause forward and backward chaining.

g2-get-backward-chaining-rules-for-variable

Returns a sequence of rules that would cause backward chaining to a variable.

Synopsis

g2-get-backward-chaining-rules-for-variable
(*variable*: class variable)
-> *result:* sequence

Argument	Description
variable	A G2 variable to test.

Return Value	Description
result	A sequence of all rules in the KB that would cause backward chaining to the specified variable.

g2-get-forward-chaining-focus-info

Gets the focal classes to which a generic rule applies.

Synopsis

g2-get-forward-chaining-focus-info
(*rule*: class rule)
-> *return-value*: sequence

Argument	Description
rule	The name of a G2 rule.

Return Value	Description
return-value	A sequence of structures identifying the classes and their associated local variables, if any, that are used in the specified rule.

Description

The classes that are returned are those identified by using the any syntax. The procedure returns one structure for each class name/local variable pair used in the rule. The order of the sequences does not necessarily correspond with the order of the focal classes in the rule.

The syntax for the return value is:

```
sequence
    structure
        (chaining-link-focus-class-name: symbol,
        chaining-link-focus-local-name-is-specified: truth-value,
        chaining-link-focus-local-name: symbol) ,
    . . .)
```

where:

- chaining-link-focus-class-name is the focal class to which the generic rule applies.
- chaining-link-focus-local-name-is-specified is true if the rule uses a local variable to refer to the focal class, or false if no local variable is used.
- chaining-link-focus-local-name is the name of the local variable used to refer to the focal class, or none if no local variable is used.

Note This procedure does not take into consideration the specification of the focal-classes of a rule.

Example

Suppose you have the following rule named rule-1:

```
for any engine e1
for any turbine t1
for any engine e2
if the min-temperature of e1 < the max-temperature of e2 and
    the horsepower of any engine < 300 and the speed of t1 > 300 and
    the weight of any engine > 12000 then post "trouble"
```

Here is how you invoke the procedure:

```
call g2-get-forward-chaining-focus-info(rule-1)
```

Here is the return value, which includes a sequence of four structures. The first three structures correspond with the classes referred to by using the local variables e1, e2, and t1. The last structure corresponds with the reference to the horsepower of any engine, which does not refer to a local variable.

```
sequence
    (structure
        (CHAINING-LINK-FOCUS-CLASS-NAME: the symbol ENGINE,
        CHAINING-LINK-FOCUS-LOCAL-NAME-IS-SPECIFIED: true,
        CHAINING-LINK-FOCUS-LOCAL-NAME: the symbol E1),
    structure
        (CHAINING-LINK-FOCUS-CLASS-NAME: the symbol ENGINE,
        CHAINING-LINK-FOCUS-LOCAL-NAME-IS-SPECIFIED: true,
        CHAINING-LINK-FOCUS-LOCAL-NAME: the symbol E2),
    structure
        (CHAINING-LINK-FOCUS-CLASS-NAME: the symbol TURBINE,
        CHAINING-LINK-FOCUS-LOCAL-NAME-IS-SPECIFIED: true,
        CHAINING-LINK-FOCUS-LOCAL-NAME: the symbol T1),
    structure
        (CHAINING-LINK-FOCUS-CLASS-NAME: the symbol ENGINE,
        CHAINING-LINK-FOCUS-LOCAL-NAME-IS-SPECIFIED: false,
        CHAINING-LINK-FOCUS-LOCAL-NAME: none))
```

g2-variable-has-backward-chaining-rules

Tests whether a variable causes backward chaining to a rule.

Synopsis

g2-variable-has backward chaining rules (*variable*: class variable)
-> *result:* truth-value

Argument	Description
variable	A G2 variable to test.

Return Value	Description
result	Returns true if *variable* has rules that would fire, false otherwise.

29

Sorting Operations

Describes procedures that sort array and list elements.

Introduction

Use the sorting system procedures to sort the elements of lists and arrays.

Note The procedures g2-sort-array and g2-sort-list supersede g2-sort.

Sorting Algorithm Implementations in G2

G2 uses two sorting algorithm implementations and chooses between them according to the type of list or array to be sorted, and depending on whether the sort procedure specifies a *key-function* and *comparison-function*, referred to below as sort functions. For the purposes of explanation, we call these two implementations "qsort" and "hsort," as follows:

- Qsort — When sorting any type of quantity-list or quantity-array (quantity-list, quantity-array, integer-list, integer-array, float-list, or float-array), and no sort function is provided, G2 uses a combination of quicksort and insertion sort. In almost every case, qsort yields the fastest sort times.
- Hsort — When performing a numeric sort on any untyped list or array (value-list, value-array, g2-list, or g2-array), or whenever either sort function is provided, G2 uses heapsort. In almost every case, hsort yields slower sort times.

Thus, in general, you should always use a quantity-list or quantity-array without a key function or comparison function to get the fastest sorting times.

However, in rare, pathological cases, qsort takes significantly longer than hsort. These cases can occur when the number of list or array elements is very large—greater than approximately 20,000 elements.

The sort time is based on the number of list or array elements, n. Hsort is always $O(n \log n)$. Qsort is almost always $O(n \log n)$. However, with qsort, the longest sort time can be as much as $O(n^2)$. The following table provides some indication of the difference in sort times. The table shows the number of seconds it takes to perform an ascending sort on a randomized array of integers of size n. Actual results vary depending on the computer speed.

n	qsort: normal case	hsort	qsort: worst case
10,000	< 1	14	8
50,000	1	83	207
500,000	25	very large	very large

As you can see, with 10,000 elements, the worst case for qsort is still smaller than hsort (8 vs. 14 seconds). With 50,000 elements, however, the worst case for qsort is two and half times slower than hsort (207 vs. 83 seconds). Keep in mind, though, that in ***almost every*** *case*, qsort is *much*, *much* faster (83 vs. 1 second).

Thus, if your application must sort lists or arrays with greater than approximately 20,000 elements, and if a significantly longer sort time is acceptable on very rare occasions, we recommend that you use a quantity-list or quantity-array and provide no sort functions. However, if very large sort times are never acceptable,

which might be the case if your application is life-critical or mission-critical, we recommend that you use an untyped list or array, or that you provide a sort function to guarantee an acceptable sort time in every case.

Of course, some calls to sorting require the use of a key function or comparison function. In such applications, the worst case will never occur.

g2-sort-array

Sorts the elements of a g2-array.

Synopsis

g2-sort-array
(*key-array*: class g2-array, *aux-array*: item-or-value,
key-function: item-or-value,
comparison-function: item-or-value, *add-args*: structure)

Argument	Description
key-array	Specifies the array to sort. This is the only required argument. Succeeding arguments can be false.
aux-array	An optional, secondary array to sort. If you are not using an auxiliary array, enter false for this argument.
key-function	A user-written function to perform some function on the key-list that you provide. If you do not provide a function, enter false for this argument.
comparison-function	A user-written function to perform some comparison function on the key array that you provide. If you do not provide a comparison function, enter false for this argument.
add-args	A two attribute structure indicating the sorting order and whether to allow other processing during the procedure.

Description

The g2-sort-array procedure sorts one or more arrays. The *key-array* is the first and only required argument. If you provide only this argument, it must be a quantitative array (quantity, float, or integer).

By default, the procedure sorts elements in descending order. You can change the sorting order to ascending by specifying the last argument, as described in Specifying Additional Arguments.

Each remaining argument can be an item or false, but only certain cases and combinations make sense. Any combination that is not permitted signals an error.

Providing the Key Array

If you provide either just the first, or the first two arguments (*key-array* and *aux-array*), key-array must be a numeric array.

Providing an Auxiliary Array

The *aux-array* is an additional array to sort, of the same length as the *key-array*, but not necessarily of the same type. For example, if *key-array* is an integer-array, *aux-array* can be a symbol-array, or any other type. Enter false if you are not using an *aux-array*.

When you specify an *aux-array* and supply no other arguments, the system procedure sorts the auxiliary elements in conjunction with the *key-array*. The result of the sort is that the auxiliary elements are in the same order as the *key-array* elements.

For example, if the key array consists of the values 2, 10, 76, and the auxiliary symbol array consists of: pineapple, apple, melon, the result of sorting these two arrays in the default descending order is as follows:

key-array values: 76, 10, 2
aux-array values: melon, apple, pineapple

Supplying a Key Function

The *key-function* must be a function definition or false. When you supply a key function, the system procedure applies it to each *key-array* element *prior* to sorting. The resulting values are then sorted, and the *key-array* and the *aux-array* (if any) are sorted according to the *key-function* values.

The *key-function* can return any value, not just floats as in the g2-sort system procedure.

Note Because the existence of a key function affects the *key-array* elements prior to sorting, if it is provided, it generates the array that is actually sorted, and the *key-array* is essentially treated as another auxiliary array.

Using a Comparison Function

The *comparison-function* is a function definition or false.

A comparison function is required when:

- You are not sorting quantitative values.
- The *key-function* is not provided and the *key-array* is not quantitative.
- The *key-function* is provided and it returns non-quantitative values when applied to the *key-array*.

The *comparison-function* is applied to pairs of values from the array being sorted and returns true or false.

Note If you provide a comparison function, sorting is significantly slower than without such a function.

Specifying Additional Arguments

The *add-args* structure has two subattributes:

Subattribute	Type	Description
direction	symbol	Either ascending or descending, indicating the desired sort order. The default is descending.
allow-other-processing	truth-value	Whether G2 enters a wait state while sorting the array.

Specifying the direction attribute as ascending changes the default sort order. Changing allow-other-processing to true allows other executable tasks to interrupt sorting as necessary.

You must provide a structure for this argument, even if the structure is empty.

Caution If you specify allow-other-processing as true, G2 enters a wait state during execution of this procedure, which allows asynchronous processing that can change the G2 context. Be sure to revalidate the context as needed after the procedure returns.

---

g2-sort-list

Sorts the elements of a g2-list.

Synopsis

g2-sort-list
(*key-list*: class g2-list, *aux-list*: item-or-value,
key-function: item-or-value,
comparison-function: item-or-value, *add-args*: structure)

Argument	Description
key-list	Specifies the list to sort. This is the only required argument.
aux-list	An optional, secondary list to sort. If you are not using an auxiliary list, enter false for this argument.
key-function	A user-written function to perform some function on the key-list that you provide. If you do not provide a function, enter false for this argument.
comparison-function	A user-written function to perform some comparison function on the key list that you provide. If you do not provide a comparison function, enter false for this argument.
add-args	A two-attribute structure indicating the sorting order and whether to allow other processing during the procedure.

Description

The g2-sort-list procedure sorts one or more lists. The key list is the first and only required argument for this procedure. If you provide only this argument, *key-list* must be a quantitative list (quantity, float, or integer).

By default, the procedure sorts elements in descending order. You can change the sorting order to ascending by specifying the last argument, as described in Specifying Additional Arguments.

Each remaining argument can be an item or false, but only certain cases and combinations make sense. Any combination that is not permitted signals an error.

Providing the Key List

If you provide either just the first, or the first two arguments (*key-list* and *aux-list*), *key-list* must be a numeric list.

Providing an Auxiliary List

The *aux-list* is an additional list to sort, of the same length as the *key-list*, but not necessarily of the same type. For example, if *key-list* is an integer-list, *aux-list* can be a symbol-list, or any other type. Enter false if you are not using an *aux-list*.

When you specify an *aux-list* and supply no other arguments, the system procedure sorts the auxiliary elements in conjunction with the *key-list*. The result of the sort is that the auxiliary elements are in the same order as the *key-list* elements.

For example, if the key list consists of the values 70.6, 81.5, 90.4, 89.7, and the auxiliary symbol list consists of: cool, medium, hot, medium, the result of sorting these two lists in the default descending order is as follows:

key-list values: 90.4, 89.7, 81.5, 70.6
aux-list values: hot, medium, medium, cool

Supplying a Key Function

The *key-function* must be a function definition or false. When you supply a key function, the system procedure applies it to each *key-list* element *prior* to sorting. The resulting values from the *key-function* are then sorted, and the *key-list* and the *aux-list* (if any) are sorted according to the *key-function* values.

The *key-function* can return any value, not just floats as in the g2-sort system procedure.

Note Because the existence of a key-function affects the *key-list* elements prior to sorting, if it is provided, it generates the list that is actually sorted, and the *key-list* is essentially treated as another auxiliary list.

Using a Comparison Function

The *comparison-function* is a function definition or false.

A comparison function is required when:

- You are not sorting quantitative values.
- The *key-function* is not provided and the *key-list* is not quantitative.
- The *key-function* is provided and it returns non-quantitative values when applied to the *key-list*.

The *comparison-function* is applied to pairs of values from the list being sorted and returns true or false.

Note If you provide a comparison function, sorting is significantly slower than without such a function.

Specifying Additional Arguments

The *add-args* structure has two subattributes:

Subattribute	Type	Description
direction	symbol	Either ascending or descending, indicating the desired sort order. The default is descending.
allow-other-processing	truth-value	Whether G2 enters a wait state while sorting the array.

Specifying the direction attribute as ascending changes the default sort order. Changing allow-other-processing to true allows other executable tasks to interrupt sorting as necessary.

You must provide a structure for this argument, even if the structure is empty.

Caution If you specify allow-other-processing as true, G2 enters a wait state during execution of this procedure, which allows asynchronous processing that can change the G2 context. Be sure to revalidate the context as needed after the procedure returns.

Example

This example sorts two lists in ascending order:

```
call g2-sort-list(key-list, aux-list, false, false,
    structure(direction: the symbol ascending, allow-other-processing: false)
```

List	Initial List Elements	Sorted List Elements
key-list	19, 5, 200, 111, 6	5, 6, 19, 111, 200
aux-list	the symbols: symbol-19, symbol-5, symbol-200, symbol-111, symbol-6	the symbols: symbol-5, symbol-6, symbol-19, symbol-111, symbol-200

The initial contents of the lists are:

```
key-list: 19, 5, 200, 111, 6
aux-list: the symbol symbol-19, the symbol symbol-5,
    [2] the symbol symbol-200,
    [3] the symbol symbol-111
    [4] the symbol symbol-200
```

g2-sort

Sorts the elements of a list.

Synopsis

g2-sort
(*key-list*: class g2-list, *aux-list*: item-or-value,
key-function: item-or-value,
comparison-function: item-or-value)

Argument	Description
key-list	Specifies the list to sort. This is the only required argument. If you provide either just the first, or the first two arguments (*key-list* and *aux-list*), then the key-list must be a numeric list of type quantity, integer, or float. The system procedure sorts numerically, in descending order, with the largest element first.
aux-list	An optional, secondary list. When you specify an *aux-list* and supply no other arguments, the system procedure sorts the aux-list elements in conjunction with the key-list. The result of the sort is that the aux-list is in the same order as the key-list, as if the elements of each list were linked. For example, if the key list has the values 2, 10, 76, and the auxiliary list has the values r, a, x, the result of passing these two lists is as follows: key-list values: 76, 10, 2 aux-list values: x, a, r

Argument	Description
key-function	A user-written function to perform some function on the key-list that you provide. For example, you could write a function that extracted the value of some item in a list and use those values as the subject of the sort procedure. Providing a key-function argument without a comparison function replaces the key-list elements with the result of the function.
comparison-function	A user-written function to perform some comparison function on the key-list that you provide. Using a comparison function is the most efficient way to perform an arbitrary sort. For example, you could write a function that compared the differences between the key-list and the aux-list. The elements of key-list would have the result of the comparison.

Note While you can still use this system procedure, it has been superseded by the procedures g2-sort-array and g2-sort-list.

Description

Sorts the list passed as the key-list argument in descending order. This procedure requires a minimum of one, and a maximum of four, arguments. Only the first argument, key-list, is required. Blank arguments are entered as false.

This system procedure does not return anything; it sorts the list or lists that you pass to it.

Caution

G2 enters a wait state during execution of this procedure, which allows other processing that can asynchronously change the G2 context. Be sure to revalidate the context as needed after the procedure returns.

Examples

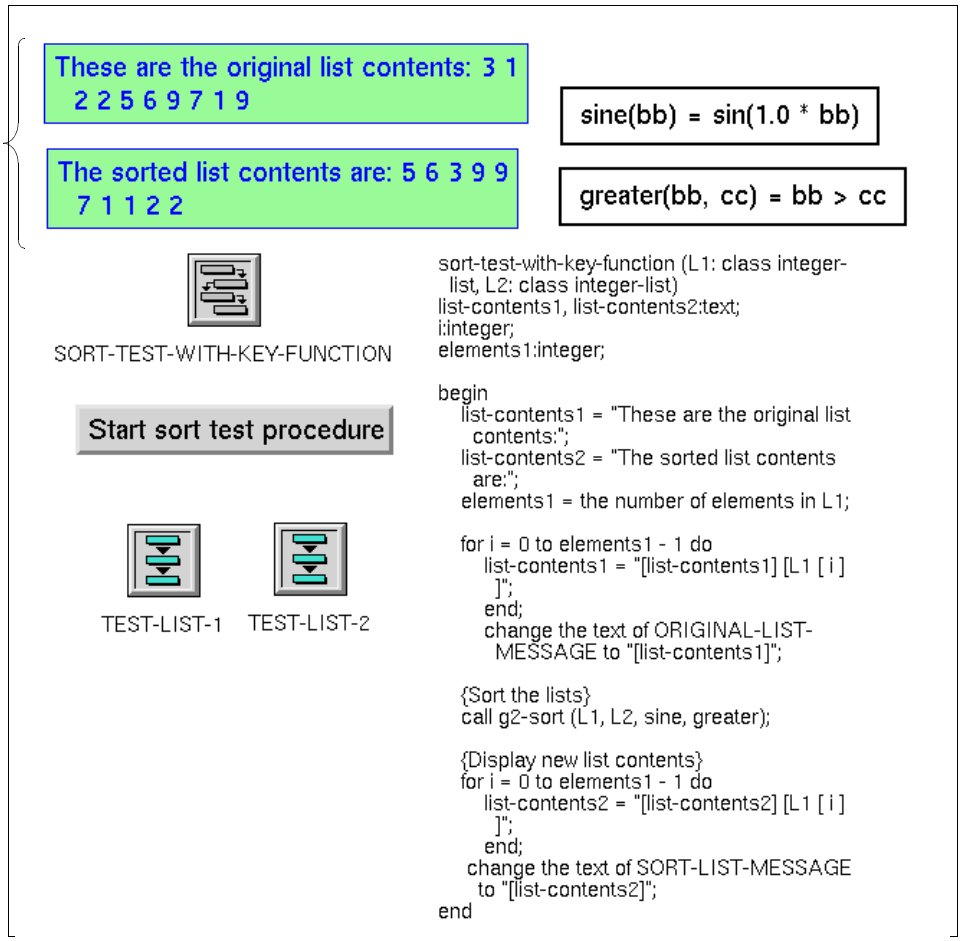
The following examples show how to use the g2-sort system procedure using one or more arguments.

The following system-procedure call has integer-list-1 as the single key-list argument. The remaining arguments are false. Given list-element values of 10, 99, 4, 212; G2 rearranges the values in descending order: 212, 99, 10, 4.

```
call g2-sort(integer-list-1, false, false, false)
```

The second example shows a procedure using g2-sort with all four arguments. The key-list and aux-list arguments are integer lists, the key-function is a user-written function sine, shown here, which uses the G2 sin function. The fourth argument, greater, is a user-written comparison function that compares the values of both the key-list and the aux-list.

When you include both key-function and comparison-function arguments, the system procedure performs the key-function first on each element of the key-list, and then the comparison function.



The key-list elements of test-list-1 are the result of these two functions.

Notice the values of the contents of the key-list, called out on the example. and the contents of that list following the g2-sort procedure. The values do not appear to be sorted. However, what appears is a list of the original values, now sorted in ascending order of their sine values.

When this example calls g2-sort, the sine function first assigns sine values to the original values, after which the greater function sorts the sine values in ascending order.

For this example, here is how the two functions work together in g2-sort on the original values:

Original Value	Sine Value	Sorted in Ascending Order
3	0.141	5 (-0.959)
1	0.841	6 (-0.279)
2	0.909	3 (0.141)
2	0.909	9 (0.412)
5	-0.959	9 (0.412)
6	-0.279	7 (0.657)
9	0.412	1 (0.841)
7	0.657	1 (0.841)
1	0.841	2 (0.909)
9	0.412	2 (0.909)

30

Sound Generation Operation

Describes the procedure that beeps the terminal displaying a specified G2 window.

Introduction

Use the sound generation system procedure to generate an audio signal from the computer.

g2-beep

Beeps in a specified window.

Synopsis

g2-beep
(*client*: class ui-client-item)

Argument	Description
client	Specify the name of a g2-window or ui-client-session item.

Example

```
call g2-beep (my-ui-client-item);
```

31

Time Information Operations

Describes procedures that access the system time and convert time-value formats.

Introduction

Use the time information system procedures to access the current time, access the time at which G2 was started, and convert a float time value to a text value in a calendar format.

The upper limit on textual timestamps that can be generated from unix-time floats, such as g2-unix-time-to-text-4-digit-year, is now Jan. 19, 2100 03:14:07 GMT. Previous, it was Jan. 19, 2038 03:14:07 GMT.

g2-text-time-interval-to-unix-time

Converts a time interval string into a float representing the number of seconds in the time interval.

Synopsis

g2-text-time-interval-to-unix-time
(*text-interval*: text)
-> *seconds*: float

Arguments	Description
text-interval	The textual representation of a time interval to convert.

Return Value	Description
seconds	The number of seconds in the interval, represented as a float.

Description

The *text-interval* is a text string in the following format, as you would enter a validity interval in a variable:

"*dd* days, *hh* hours, *mm* minutes, and *ss* seconds"

Example

This example uses the system procedure to convert a text string to a float:

```
float-value = call
    G2-text-time-interval-to-unix-time("1 hour, 30 minutes, and 30 seconds")

    --> 5430.0
```

g2-text-time-stamp-to-unix-time

Converts a timestamp string to a float representing the number of seconds between the UNIX base time: midnight Jan 1, 1970 ("01 jan 1970 12:00 a.m.") and the time specified by the timestamp.

Synopsis

g2-text-time-stamp-to-unix-time
(*text-time-stamp*: text)
-> *seconds*: float

Arguments	Description
text-time-stamp	The timestamp text to convert.

Return Value	Description
seconds	The number of seconds represented as a float.

Description

The *text-time-stamp* is a text string in the same format as the time stamp that appears in the authors attribute, without the parentheses:

"*dd mon yyyy hh:mm a.m.*"

Timestamp element	Description
dd	The two-digit day of the month, for example 17, prefixed with a 0 if the day is less than 10.
mon	The three-character month of the year which can be: jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, or dec.
yyyy	The four-digit number of the year, for example: 1997.

Timestamp element	Description
hh:mm	The hour and minute of the day, separated by a colon, for example: 10:45. Use 12:00 a.m. for midnight and 12:00 p.m. for noon.
a.m.	The before or after noon specifier, which can be a.m. or p.m.

Example

This example uses the system procedure to convert a text timestamp to a float value:

```
float-value = call g2-text-time-stamp-to-unix-time("16 Jan 2000 12:35 p.m.")
    --> 9.48e8
```

g2-unix-time

Returns the current system time.

Synopsis

g2-unix-time
( )
-> *time*: float

Return Value	Description
time	A float of the current time.

Description

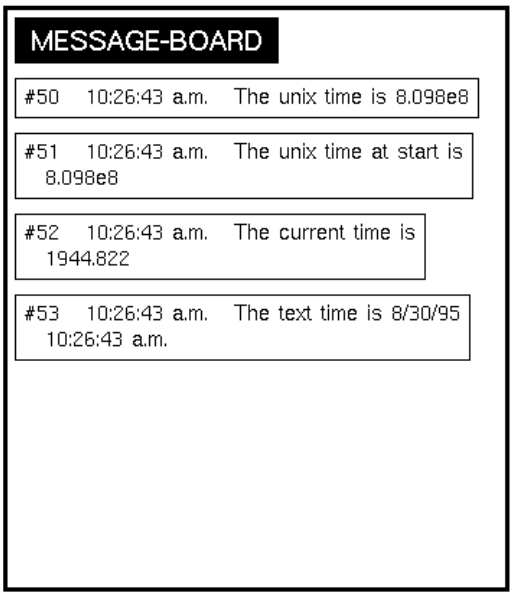
UNIX systems account for Daylight Savings Time automatically. Also, g2-unix-time returns the number of seconds that have elapsed since midnight, January 1, 1970, Greenwich Mean Time.

Example

The following procedure illustrates the use of all the time information system procedures:

```
unix-time( )
    unix-time: float;
    unix-time-at-start: float;
    text-time: text;
    current-time: float;
begin
    unix-time = call g2-unix-time( );
    inform the operator that "The unix time is [unix-time]";

    unix-time-at-start = call g2-unix-time-at-start( );
    inform the operator that "The unix time at start is [unix-time-at-start]";

    current-time = unix-time - unix-time-at-start;
    inform the operator that "The current time is [current-time]";
    text-time = call g2-unix-time-to-text(unix-time, true);
    inform the operator that "The text time is [text-time]";
end
```

The procedure unix-time( ) displays messages such as the following:



g2-unix-time-at-start

Returns the time when G2 was started most recently.

Synopsis

g2-unix-time-at-start
( )
-> *g2-start-time*: float

Return Value	Description
g2-start-time	A float value of the time when G2 was most recently started.

Description

G2-unix-time-at-start returns the time when G2 was most recently started. The time is expressed as the number of seconds that had elapsed since January 1, 1970, Greenwich Mean Time, at the moment when G2 was started.

If you subtract this time from the return value of g2-unix-time, and then round it, it returns the time in *the current time* format.

Example

See the example for g2-unix-time.

g2-unix-time-to-text-4-digit-year

Converts a float representing Unix time to a time string with a 4-digit year designator.

Synopsis

g2-unix-time-to-text-4-digit-year
(*unix-time*: float, *use-seconds*: truth-value)
-> *time-as-text*: text

Attribute	Description
unix-time	The UNIX time. You can use the g2-unix-time system procedure to return the time.
use-seconds	If true, add seconds to the end of the returned text, such as 9/30/2005 4:45:07 p.m. If false, does not add seconds to the end of the returned text, as: 9/30/2005 4:45 p.m.

Return Value	Description
time-as-text	A text string of the local time.

Description

Converts *unix-time* to a calendar time in the format:

mm/dd/yyyy hh:mm {a.m. | p.m.}

where mm, dd, and yyyy are the month, day, and year, respectively.

Note The g2-unix-time-to-text system procedure, which returns a 2-digit year, has been superseded by g2-unix-time-to-text-4-digit-year.

Example

```
float-value = g2-unix-time( ):
text-value = g2-unix-time-to-text-4-digit-year(float-value, true)
    --> "11/27/2002 11:16:34 a.m."
```

g2-get-current-time-zone

Get current time zone name from underlying operating system.

Synopsis

g2-get-current-time-zone
    ()
    -> *time-zone-name*: text, *bias*: integer

Return Value	Description
time-zone-name	A text string of the current time zone.
bias	The current bias for local time translation on this computer, in minutes.

Description

Get current time zone name and bias from underlying operating systems. Daylight time zone names are supported by this procedure.

On Windows, the procedure will call Win32 API GetTimeZoneInformation() to get the need information, and in this case the result names were in its long form, like this:

```
GMT Daylight Time
```

and on Linux and other Unix-based systems the procedure will call Unix system call tzname() instead, and in such cases the returning value should be just a few upper-case letters.

The bias is the difference, in minutes, between Coordinated Universal Time (UTC) and local time. All translations between UTC and local time are based on the following formula:

```
UTC = local time + bias
```

Example

```
text-value = g2-get-current-time-zone()
    --> "GMT Daylight Time", 0
text-value = g2-get-current-time-zone()
    --> "CST", -480
```

33

User and Security Information Operations

Describes the system procedures that allow you to access and change G2 user and security data.

Introduction

Use the user and security information operations to add and delete G2 users, set and change passwords, and obtain license and environment-variable information.

g2-add-user

Adds a new user to the OK file that authorizes the G2 process.

Synopsis

g2-add-user
(*user-name*: symbol, *clear-password:* text, *modes:* sequence,
password-expiration-date: sequence)

Argument	Description
user-name	A symbol to be used in the User name field of the Login Dialog
clear-password	A text value to be used in the Password field of the Login Dialog
modes	A sequence of symbols naming user modes
password-expiration-date	A sequence of three integers that indicate, in order, the day, the month, and the year; or a sequence with a single zero component, sequence(0), indicating that the password never expires

Description

If the arguments are valid, this procedure changes the OK file to include a new user with the given password, user modes, and expiration date.

This procedure generates an error for these conditions:

- *user-name* already names an authorized user.
- *modes* is the empty sequence, has a non-symbolic component, or includes proprietary which is an invalid OK file user mode.
- *password-expiration-date* is earlier than the current date.

Example

```
g2-add-user(the symbol mdavis, "tripwith",
    sequence(the symbol developer, the symbol administrator),
    sequence(3, 1, 2001))
```

g2-change-password-expiration-date

Changes the password expiration date for an authorized user in the OK file for the G2 process.

Synopsis

g2-change-password-expiration-date
(*user-name*: symbol, *day:* integer, *month:* integer, *year:* integer)

Argument	Description
user-name	A symbol naming an authorized user .
day	An integer indicating the day the password expires.
month	An integer indicating the month the password expires.
year	An integer indicating the year the password expires.

Description

If *user-name* names an authorized user and *day,month,* and *year* indicate the current day or later, this procedure changes the OK file to the new password expiration date.

G2 generates an error if *user-name* is not in the OK file, or if the expiration date is earlier than the current day.

g2-delete-user

Deletes a user from the OK file that authorizes the G2 process.

Synopsis

g2-delete-user
(*user-name*: symbol)

Argument	Description
user-name	A symbol naming an authorized user.

Description

G2 deletes a user by creating a temporary file in the directory that contains the current OK file, and then renaming the temporary file to `g2.ok`. The contents of the new OK file are the same as the old OK file except that the information for *user-name* is replaced with a blank line.

G2 signals an error if *user-name* does not name an authorized user or if the OK file directory is not writable.

g2-floating-client

Tells you whether the license for your interaction with G2 is a floating license.

Synopsis

g2-floating-client
(*client*: class ui-client-item)
-> *return-value*: truth-value

Argument	Description
client	An instance of the ui-client-item class: either a g2 window or a ui-client-session item.

Return Value	Description
return-value	true if the license is floating, false otherwise.

Description

This procedure returns true if the license associated with your g2 window or ui-client-session item is a floating license; otherwise it returns false.

g2-get-environment-variable

If the environment-variable argument is defined, this procedure returns its value.

Synopsis

g2-get-environment-variable
(*environment-variable*: text)
-> *return-value*: text

Argument	Description
environment-variable	A text value that specifies the name of an environment variable.

Return Value	Description
return-value	If *environment-variable* names a defined variable, its value is returned as a text value; otherwise the empty text value is returned.

Example

This procedure statement:

```
return-text-value = call g2-get-environment-variable("EDITOR")
```

assigns the value "/user/dt/bin/dtpad" to return-text-value.

g2-get-floating-licenses-remaining

Returns the number of licenses that have not already been used to authorize your Telewindows or Telewindows2 Toolkit connections.

Synopsis

g2-get-floating-licenses-remaining
(*client*: class ui-client-item)
-> *number-remaining*: integer

Argument	Description
client	An instance of the ui-client-item class: either a g2 window or a ui-client-session item.

Return Value	Description
number-remaining	An integer indicating the number of licenses still available.

Description

When given a g2-window item, this procedure returns the number of Telewindows licenses still available; and when given a ui-client-session item, it returns the number of Telewindows2 Toolkit licenses still available.

When running G2 without a window by including the `-no-window` command-line option and when G2 is not embedded, G2 allows one additional floating Telewindows license, which is included in the return value.

g2-get-item-with-uuid

Returns an item with a given UUID.

Synopsis

g2-get-item-with-uuid
(*uuid*: text)
-> *item*: item-or-value

Argument	Description
uuid	The text of the uuid attribute of the item to get.

Return Value	Description
item	The item with the corresponding UUID.

Description

If no item exists with the given UUID, the system procedure returns false.

g2-get-modes-for-authorized-user

Returns information about the allowable modes for a given user. This system procedure is only applicable at secure G2 sites whose OK file includes user entries.

Synopsis

g2-get-modes-for-authorized-user
(*ui-client-user-name*: symbol)
-> *user-information*: sequence

Attribute	Description
ui-client-user-name	The name of a user, given as a symbol.

Return Value	Description
user-information	If the *ui-client-user-name* passed to the procedure is located in the `g2.ok` file most recently loaded into G2, the return sequence consists of one or more symbols, each symbol representing an allowable mode for the given user. If the user name is not found in the OK file, the sequence is returned with no values.

g2-get-window-license-type

Returns information on whether the license associated with your window is a floating license or a dedicated license.

Synopsis

g2-get-window-license-type
(*client*: class ui-client-item)
-> *license-type*: symbol

Attribute	Description
client	An instance of the ui-client-item class: either a g2-window item or a ui-client-session item.

Return Value	Description
license-type	Can be either the symbol floating or the symbol dedicated.

Description

This system procedure tells you whether you are using a floating license or a dedicated license to connect to G2.

g2-make-uuid

Returns a universal unique identifier (UUID), which is unique not only to the KB, but also is *universally unique*: the ID is not duplicated in any KB, anywhere in the world, at any time, past or future, unless the KB file is copied or the same KB is loaded into another G2.

Synopsis

g2-make-uuid
( )
-> *uuid*: text

Return Value	Description
uuid	A UUID as a text.

Description

A UUID has three parts:

1. The machine ID (MachID) of the computer where the ID is generated. This is the same machine ID shown in the title block when G2 is first loaded.
2. The current time (CTime), expressed as the number of seconds to the millisecond, since January 1, 1970, Greenwich Mean Time, produced by the system procedure g2-unix-time.
3. A integer (Int) between 0 and 9999, initially random, which is incremented before generating a new UUID. When Int reaches 9999, it is reset to 0.

These parts are concatenated as a text string to form the UUID:

```
UUID = "[MachID]-[CTime]-[Int]"
```

The resulting text is approximately 28-30 characters in length. The memory allocated to carry the ID in an attribute is roughly the same number of bytes. Because of the third component of the UUID, you can create up to 10,000 UUIDs on the same machine in the same millisecond, without duplicating a UUID. (On an HP712/60 workstation or 100 MHz Pentium machine, it takes approximately a millisecond to generate one UUID, so in practice, there is a safety factor of approximately 10,000.)

g2-register-login-handler

Registers a user-defined procedure that runs when certain login events occur.

Synopsis

g2-register-login-handler
(*handler*: item-or-value)
-> *registration-successful*: truth-value

Attribute	Description
handler	The user-defined procedure that is to be called when certain login events occur.

Return Value	Description
registration-successful	Returns true if the procedure is successfully registered; otherwise, returns false. This procedure also returns false passing false as the *handler* to deregister the handler.

Description

This system procedure registers a login handler to be run whenever a user attempts to log into your secure G2. The login handler is a G2 procedure you write that specifies the actions you want performed when successful or failed login events occur. Changing a user's password is not considered a login event and will not call your login handler.

You can pass false to the g2-register-login-handler procedure to deregister the current login handler. The procedure returns false in this case.

Writing the Login Handler

The login handler must accept a structure as an argument. The structure is returned by the system login function and has the following attributes:

Attribute	Value
success	The symbol true if the login succeeded, false otherwise.
system	The symbol tw for Telewindows or t2 for Telewindows2 Toolkit.
status	A symbol describing the event.
user-name	A symbol.
user-mode	A symbol.
network-info	The `icp-connection-name` string for connections over the network and false otherwise.

The `icp-connection-name` string provides information about the protocol of the connection and the hostname of the machine attempting to connect.

The login handler may use this information in any way necessary. The following example shows a login handler that simply prints the information in the structure to the message board.

```
default-login-handler(info-structure: structure)
message-text: text;
begin
    if (the success of info-structure)
        then message-text = "succeeded"
        else message-text = "didn't happen [the status of info-structure];
    post "Login [message-text] in system [the system of info-structure]
        for user: [the user-name of info-structure]
        in mode: [the user-mode of info-structure]
        from [the network-info of info-structure]."
end
```

Example

This example calls the system procedure and assigns the returned value to a truth-value variable:

```
success = call g2-register-login-handler(default-login-handler)
```

g2-reinstall-authorized-users

Directs G2 to reread the OK file for the G2 process and install the authorized users.

Synopsis

g2-reinstall-authorized-users
( )

Description

If G2 does not encounter any syntax errors in the OK file, it reinstalls the authorized users; otherwise it signals an error.

g2-set-maximum-login-attempts

Sets the maximum number of unsuccessful logins that a secure G2 tolerates before it shuts down a G2 window.

Synopsis

g2-set-maximum-login-attempts
(*max-logins*: integer)

Argument	Description
max-logins	An integer that indicates the limit of unsuccessful logins G2 processes before closing a connection.

Description

This procedure allows you to change the default number of login attempts allowed before a connection to a secure G2 is closed. The default value is 5.

Example

The following call directs G2 to close a connection after two unsuccessful logins have been attempted:

```
g2-set-maximum-login-attempts(2)
```

g2-set-user-password

Sets a user password programmatically without stopping G2 execution.

Synopsis

g2-set-user-password
(*user-name*: symbol, *old-password*: text, *new-password*: text)
-> *result*: symbol

Attribute	Description
user-name	The name of the G2 user as a symbol.
old-password	The current password for the user.
new-password	The new password for the user.

Return Value	Description
result	A symbol that describes the result of calling the procedure. The possible values are listed below.

Description

Sets a new password for a user who is already authorized in the OK file that was used when G2 was launched.

Caution The passwords specified to g2-set-user-password are in plain text, and can therefore be read by anyone. Such specification is a potential security risk. Be careful that passwords given to this procedure are kept secure.

If no error occurs, the procedure returns the symbol OK. If an error occurs, it returns one of the following symbols:

- unknown-user-or-bad-password
- new-password-must-be-at-least-4-characters-long
- new-password-too-long
- new-password-has-invalid-characters
- cannot-find-g2-ok-file
- problem-saving-password-in-g2-ok-file

Caution

G2 enters a wait state during execution of this procedure, which allows other processing that can asynchronously change the G2 context. Be sure to revalidate the context as needed after the procedure returns.

Authorizing Multiple Users at One Time

You can use g2-set-user-password to authorize a group of users at once. Assume that the users have all been entered in the OK file, each with the empty string as a password and that this information has then been extracted into a file with one user name per line. You can then use g2-set-user-password to give a standard password to each user in the file. Each user can later change the standard password by hand from inside of G2. The following example reads a list of users from a file and assigns each user the same password.

```
fix-passwords( )
file: class g2-stream;
user-name: text;
begin
    file = call g2-open-file-for-read("/home/abc/user-names.txt");
    repeat
        user-name = call g2-read-line(file);
        exit if the g2-stream-status of file = the symbol end-of-file-reached;
        call g2-set-user-password(symbol("[user-name]"), "", "new-password");
    end;
    call g2-close-file(file)
end
```

g2-validate-user-and-password

Tells you whether the user name and password are valid.

Synopsis

g2-validate-user-and-password
(*user-name*: symbol, *clear-password*: text)
-> *return-value*: symbol

Arguments	Description
user-name	The user-name symbol you wish to validate.
clear-password	The password text you wish to validate.

Return Value	Description
return-value	If *user-name* and *clear-password* are valid, this procedure returns the symbol ok; otherwise it returns one of these error symbols: unknown-user-or-bad-password or password-has-expired.

Example

The symbol unknown-user-or-bad-password is returned for this example system procedure call:

```
return-value-symbol = call g2-validate-user-and-password
                                    (the symbol bogus-name, "xyzmtt")
```

Caution This system procedure requires G2 to be running in secure mode and will always succeed otherwise.

34

User Interface Operations

Describes procedures for creating and managing Windows user interface components in the Telewindows client, and for controlling workspace and item appearance, obtaining information about events, and performing other user interface operations.

Introduction

Use the user interface system procedures to manipulate items and change their appearance. The user interface procedures are separated into these categories:

- Chart views

- Dialog views
- Editor operations
- Graphics-oriented operations
- HTML views
- Input-handling operations
- Native Menu System
- Parsing operations
- Property grid views
- Reflection, rotation, size, and layering operations
- Selection API
- Shortcut bar views
- Status bar operations
- System commands
- Text operations
- Tree views
- Trend-chart operations
- Window and workspace operations
- Window handles and views

You can use the item-layering system procedures in conjunction with the paint mode drawing facilities. The procedures manipulate item layer positions upon a workspace, and provide functionality corresponding to the item menu choices, lift to top and drop to bottom.

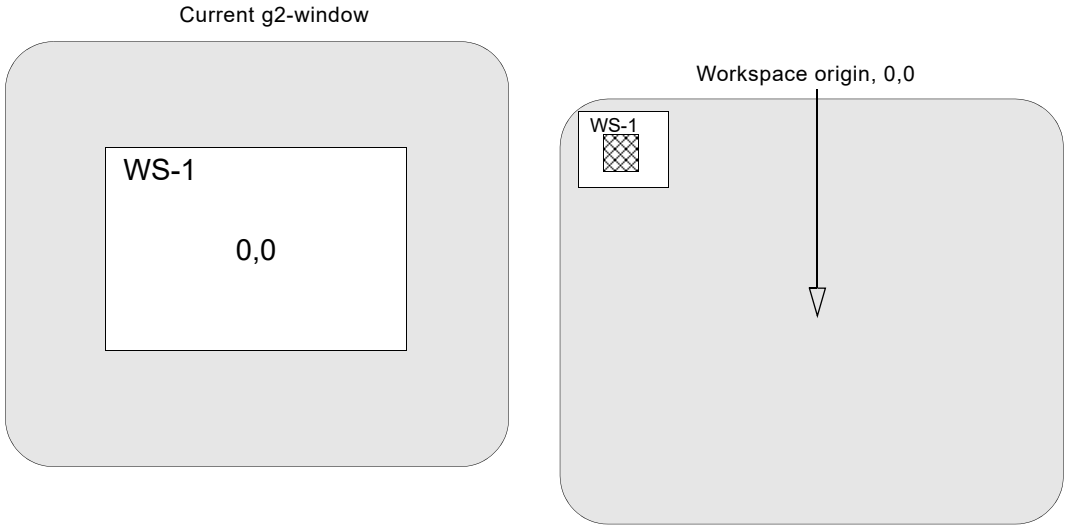
Other user-interface operations let you control certain drawing and refresh capabilities of your KB.

Positional Coordinates of Items

All G2 items are positioned according to their x and y coordinates. Within every KB, there are actually two separate coordinate systems at work — the x and y coordinates of items that reside upon workspaces, and the x and y coordinates of workspace items within the G2 window.

The x and y coordinates of items upon a workspace are measured in relation to the origin of the workspace, which is the original center, the item x, y position of 0, 0 as shown in the first diagram. The **workspace origin** can change as the

workspace changes size, so that it is no longer in the center, as indicated in the second diagram showing ws-1 after it has been shrinkwrapped.



Animation expressions exist to reference the x, y coordinates of items upon workspaces, the item-x-position, and the item-y-position.

G2 measures the x, y coordinates of workspaces within a window in relation to the center of the G2 window, which is also the x, y position of 0, 0. The center of the G2 window remains constant.

You can use system procedures to convert the x, y coordinates of items upon workspaces into coordinates within the G2 window, and the x, y coordinates of workspaces within the G2 window into item coordinates.

Size and Scale of Items

In addition to having an x, and y position upon a workspace or, for workspaces, within a G2 window, items also have an associated size and scale.

The size of all items is measured in workspace units as item-height and item-width. Animation expressions return these values (the item-height of item and the item-width of item). The item menu choice change-size lets you enlarge or decrease the size of any item upon a workspace. A workspace increases its size when you drag items away from the center in any direction and push the workspace border outwards. You decrease the size of a workspace by using the shrinkwrap action or menu choice.

You can only scale workspace items. However, changing the scale of a workspace affects the scale of all items that reside upon that workspace. By using the Control + b (bigger) command to enlarge a workspace, or Control + s (smaller) command to decrease its size, or showing a workspace at a particular scale, G2 simultaneously scales all items upon the workspace proportionately.

You can scale workspaces up to four times larger than full scale, using the Control + b command successively. G2's scaling lets you display large items at a small scale, or small size items at a large scale.

System procedures return the current x and y scale of any workspace within a g2-window.

Item Layering and Connections

Item layering for non-workspace items — lifting items to the top or dropping them to the bottom of the current item-layering order—has a particular effect on items with connections.

The item-layer of connections is determined by the item from which the connection begins. For example, given two items A and B, with item layer numbers 1 and 2, respectively, drawing a connection from A to B assigns the layer-number 1 to the connection.

The item-layer is not related to the connection direction. That is, joining an output connection to an input stub (the opposite direction of the input connection), still assigns the layer number of the output stub item to the entire connection. For more information about connections and their item layer numbers, see Connections in the *G2 Reference Manual*.

Chart Views

G2 provides a Windows chart view, which you access through Telewindows.

Note Chart views are only supported in Telewindows Next Generation (`twng.exe`).

For examples, see Using Chart Views in Windows Views, Panes, and UI Features in the *G2 Reference Manual*.

g2-ui-create-chart-view
(*title*: text, *callback*: symbol, *options*: structure, *win*: class g2-window)
-> *handle*: integer

Creates a chart view as an MDI child window with the given *title* on *win*, and returns an integer handle to the view. If the window does not support chart views, or if the required DLL cannot be found on the Telewindows machine, an error is signalled.

The *callback* is a procedure that gets called when the user closes the chart view or, if enabled, when the user clicks on a data point. Specify the callback as the symbol none to provide no callback. The syntax of the callback procedure is:

my-chart-view-callback
(*event*: symbol, *win*: class g2-window, *handle*: integer, *data*: value, *info*: structure, *user-data*: value)

Argument	Description
event	A symbol naming the event that invoked the callback. The options are: click or closed. **Note:** To receive click events, you must specify allow-data-hot-spots as true in the chart properties. The callback also responds to the event types described in View Events.
win	The window in which the chart view appears.
control	The integer handle of the chart view.
data	For click events, a structure with this syntax: structure (x: *float*, y: *float*, z: *float*)

Argument	Description
info	For click events, a structure with this syntax: structure (subset: *integer*, point: *integer*) where: • subset is the curve number, which is numbered from 0. It is the same as the *i* argument to the g2-ui-chart-set-data procedure. • point is the point number with that curve or subset, also numbered from 0. For a description of the *info* structure for the closed event, see the description of the callback for g2-ui-create-html-view. The state attribute in the *info* structure returns normal.
user-data	For the closed event, the symbol none. For the click event, the symbol user-data.

The *options* argument is a structure with this syntax or, to support duplicate attributes, a sequence of structures that are guaranteed to be applied in order:

```
structure
(type: symbol,
 left: integer,
 top: integer,
 width: integer,
 height: integer,
 container: symbol-or-integer,
 neighbor: symbol,
 dock: symbol)
```

where:

- type is one of these symbols: graph (default), pie, scigraph or sgraph, polar, and 3d
- left, top, width, height define the position and size of the MDI child window.
- container is one of:
  - mdi-child, as a symbol, which displays the view in a floating pane. The default is mdi-child.
  - pane, as a symbol, which displays the view in a docked pane.

  - A dialog handle, as an integer, which displays the view in a dialog. The view is placed in the dialog, replacing the existing control, which must be a label. When container is a dialog handle, the neighbor attribute can be the control ID of another dialog against which to dock, in which case the dock attribute must be the symbol within.
  - The handle of a listbar-style shortcut bar, in which case the neighbor option is the number of the folder within the listbar into which to create the view.

- neighbor — Another pane against which to dock, as a view designation. See g2-ui-is-valid-window-handle for a description of a view designation. The default value is the overall frame.

  To specify that the view pane be placed in a tab pane within another pane, specify these options: dock: the symbol within, neighbor *h*, where *h* is the pane within which to place the view pane. When container is a dialog handle, the neighbor attribute can be the control ID of another dialog against which to dock.

- dock — The docking edge, which can be one of these symbols: left, top, right, bottom, float, or within. The default value is left.

See Appendix B, Chart Properties and Enumeration Values for a list of the known chart properties and a list of the known property values, for properties whose values are an enumeration.

The procedure returns an integer view handle.

g2-ui-modify-chart-view
(*handle*: integer, *properties*: structure, *win*: class g2-window)

Modifies the given properties of the chart view denoted by *handle* on the given g2-window. Note that you must update the chart for the changes to fully take effect.

The *properties* argument is a structure that provides the chart properties to modify, or, to support duplicate attributes, a sequence of structures that are guaranteed to be applied in order. See Appendix B, Chart Properties and Enumeration Values.

g2-ui-chart-set-data
(*handle*: integer, *property*: symbol, *i*: integer, *j*: integer, *val*: value, *win*: class g2-window)

Sets a single element in a 1D array-valued property, such as subset-colors, or a 2D array-valued property, such as xdata, ydata, and zdata.

For 1D properties, the *i*-th element is set, and *j* is ignored. For the 2D properties, xdata, ydata, and zdata, *i* is the subset or curve number and *j* is the point number, both of which are numbered from 0.

g2-ui-manage-chart-view
(*action*: symbol, *handle*: integer, *arg*: value, *win*: class g2-window)

Performs an action on the chart view denoted by *handle*. The options for *action* are:

- update — Updates the chart window to show any recent property changes.
- partial-update — Redraws a set of points, for all subsets, on the chart view. This action does not update any other properties of the chart view; you should still perform a complete update periodically by using the update action.
- destroy — Destroys the chart view.
- print — Posts a print dialog for the chart view.
- copy — Copies the chart image as an enhanced metafile to the clipboard.
- export — Exports the chart to a client-side file as a bitmap or a Windows image file.

When *action* is partial-update, *arg* is a structure with the following syntax:

structure
(start: *integer*
 count: *integer*)

where:

- start is the index of the first data point to update. The default value is 0.
- count is the number of data points to update. The default is 1.

When *action* is export, *arg* can be a text or a structure, as follows:

- text — The pathname of a file on the client.

  or

- structure
  (pathname: *text*,
   width: *integer*,
   height: *integer*,
   format: *symbol*)

  where:

  - pathname — The pathname of a file on the client.
  - width — The width, in pixels, of the exported image. The default is the width of the view.
  - height — The height, in pixels, of the exported image. The default is the height of the view.

- format — The format of the image file, which is one of these symbols: JPEG, PNG, BMP, or WMF. The default is computed from the file type of the pathname.

Dialog Views

You can use the dialog API procedures to launch and access various dialogs that are built into the windowing system on which Telewindows is running, as well as custom dialogs on Windows platforms. These dialogs correspond closely with the system-defined dialogs that GUIDE provides. Currently, the only windowing system that supports these dialogs is Microsoft Windows.

The standard dialog API supports these types of dialogs:

- Custom dialogs
- Miscellaneous dialogs:
  - Basic
  - Query
  - File
  - Print
- Notification dialogs
  - Notification
  - Delay notification

For information on system-defined GUIDE dialogs, see Chapter 8 in the *G2 GUIDE User's Guide*.

Note The system procedures for "modal" dialogs, that is, those that return a value, create and delete a new symbolic or text parameter during their execution. This could possibly interfere with existing KBs.

To access the standard dialog API procedures, choose G2 System Procedures > g2-user-interface-operations > dialog-api to display this workspace:

Custom Dialogs

You use these system procedures to post, modify, and query custom Windows dialogs.

For examples, see Windows Dialogs in the *G2 Reference Manual*.

For detailed information on the arguments to these procedures, see Custom Windows Dialogs in the *G2 Reference Manual*.

g2-ui-post-custom-dialog
(*dialog-specification*: structure, *user-data*: value, *win*: g2-window)
-> *dialog-handle*: integer

Posts a dialog on a remote client's window. This procedure returns an integer that is a handle to the posted dialog. It signals an error if the specified window does not support standard Windows.

Argument	Description
dialog-specification	A structure that describes the dialog. For details, see Dialog Specification in Custom Windows Dialogs in the *G2 Reference Manual*.
user-data	An item or value passed to the callback when it is invoked. For example, you can use the same callback for multiple registrations and specify different *user-data* to differentiate each callback.
win	The g2-window on which to post the dialog. This window must be running on a Windows machine.

Return Value	Description
dialog-handle	An integer that provides a handle to the custom dialog that gets created.

g2-ui-modify-custom-dialog
(*dialog-handle*: integer, *modify-specification*: sequence, *win*: class g2-window)

After a dialog has been posted, changes the contents of a dialog.

Argument	Description
dialog-handle	The dialog handle that is returned by the g2-ui-post-custom-dialog system procedure.
modify-specification	A sequence of structures that describe the dialog components to modify. For details, see Modify Specification in Custom Windows Dialogs in the *G2 Reference Manual*.
win	The g2-window on which to post the dialog. This window must be running on a Windows machine.

g2-ui-query-custom-dialog-values
(*dialog-handle*: integer, *controls*: sequence, *win*: class g2-window)
-> *control-values*: sequence

After a dialog has been posted, queries the contents of the dialog.

Argument	Description
dialog-handle	The dialog handle that is returned by the g2-ui-post-custom-dialog system procedure.
controls	This argument is currently not used.
win	The g2-window on which to post the dialog. This window must be running on a Windows machine.

Return Value	Description
control-values	A sequence of current control values for all controls in the dialog.

Note For a tabular view, the *new-value* argument to the dialog update callback returns only the selected-rows of the control-value structure, not the columns and rows. The reason is that the data cannot be changed by the user, and there may be many rows of data. Additionally, the architecture is such that the model and view are kept separate, and a separate controller is responsible for handling callbacks from the view and updating the model. To modify the view, you use the g2-modify-custom-dialog system procedure. For an example, see the description of the Alert Queue Demo in Chapter 9 "Creating Windows Dialogs" in the *Telewindows User's Guide*.

g2-ui-accept-custom-dialog
(*dialog-id*: integer, *win*: class g2-window)

Simulates the user clicking the OK button in a custom dialog displayed on a window. The *dialog-was-accepted* argument in the dialog-dismissed-callback is true.

g2-ui-cancel-custom-dialog
(*dialog-id*: integer, *win*: class g2-window)

Dismisses a custom dialog displayed on a window. The *dialog-was-accepted* argument in the dialog-dismissed-callback is false.

g2-ui-cancel-dialogs
(*window*: class g2-window)

Cancels all dialogs that are currently posted on *window*. Dialogs that return results act as if their Cancel button was pressed. Dialogs that do not return results are simply discarded. This API procedure is called automatically when the KB is reset or cleared.

This procedure cancels all dialogs in the current window after a delay:

CANCEL-DIALOGS-AFTER-DELAY

```
cancel-dialogs-after-delay(win: class g2-
  window)
begin
 wait for 5 seconds;
 post "Cancelling@.@.@.";
 call g2-ui-cancel-dialogs(win);
end
```

g2-ui-grid-view-insert-column
(*dialog*: integer, *control*: value, *column*: integer, *column-spec*: structure, *win*: class g2-window)

Inserts a column into the grid-view specified by *control*, which is the control-id of the control, in *dialog*. The *column* is the *column-id* of the new column, after it has been inserted. The *column* argument must not be greater than the number of columns in the grid after the new column has been added. For example, if the grid has columns 0 - 4 (5 columns), the *column* argument can be 0 - 5, inclusive. A column number of -1 means to insert as the last column.

For a description of *column-spec*, see grid-view in Custom Windows Dialogs in the *G2 Reference Manual*. The *column-spec* must specify the same number of rows as the existing columns in the grid.

g2-ui-grid-view-delete-column
(*dialog*: integer, *control*: value, *column*: integer, *win*: class g2-window)

Deletes a column from the grid-view specified by *control*, which is the control-id of the control, in *dialog*. The *column* is the *column-id* of the column to delete, which must be a valid column in the grid. A column number of -1 means to delete the last column.

g2-ui-grid-view-insert-row
(*dialog*: integer, *control*: value, *row*: integer, *row-spec*: structure, *win*: class g2-window)

Inserts a row into the grid-view specified by *control*, which is the control-id of the control, in *dialog*. The *row* is the *row-id* of the new row, after it has been inserted. The *row* argument must not be greater than the number of rows in the grid after the new row has been added. For example, if the grid has rows 0 - 4 (5 rows), the *row* argument can be 0 - 5, inclusive. A row number of -1 means to insert as the last row.

For a description of *row-spec*, see grid-view in Custom Windows Dialogs in the *G2 Reference Manual*. The *row-spec* must specify the same number of columns as the existing rows in the grid.

g2-ui-grid-view-insert-rows
(*dialog*: integer, *control*: value, *rows*: sequence, *win*: class g2-window)

Inserts several rows into the grid-view specified by *control*, which is the control-id of the control, in *dialog*. The *rows* is a sequence of *row-spec*.

g2-ui-grid-view-update-rows
(*dialog*: integer, *control*: value, *rows*: sequence, *win*: class g2-window)

Update several rows in the grid-view specified by *control*, which is the control-id of the control, in *dialog*. The *rows* is a sequence of *row-spec*.

g2-ui-grid-view-delete-row
(*dialog*: integer, *control*: value, *row*: integer, *win*: class g2-window)

Deletes a row from the grid-view specified by *control*, which is the control-id of the control, in *dialog*. The *row* is the *row-id* of the row to delete, which must be a valid row in the grid. A row number of -1 means to delete the last row.

g2-ui-grid-view-replace-cell
(*dialog*: integer, *control*: value, *row*: integer,
column: integer, *cell-spec*: structure, *window*: class g2-window)

Replaces the cell at the given *row* and *column* positions in a grid-view specified by *control*, which is the control-id of the control, in *dialog*. You can replace all grid view attributes of all control types, using the following syntax for *cell-spec*:

```
structure
(cell-type: the symbol cell-type,
cell-value: structure ( [ attribute: value ] ... ) )
```

For example, to modify the background color and text value of a text-box control in a grid view, the *cell-spec* would look like this:

```
structure
(cell-type: the symbol text-box,
cell-value:
    structure (text-value: “New Text”, background-color: the symbol blue) )
```

Note that the g2-ui-grid-view-replace-cell system procedure replaces the entire cell contents with the specified *cell-spec*, which requires that you specify the entire cell contents to replace.

g2-ui-grid-view-modify-cell
(*dialog*: integer, *control*: value, *row*: integer,
column: integer, *cell-spec*: structure, *window*: class g2-window)

Modifies only the specified attributes of a cell, leaving the unspecified attributes unchanged.

Miscellaneous Dialog Procedures

For examples, see Windows Dialogs in the *G2 Reference Manual*.

File Dialog

This API procedure posts a file dialog that allows the user to choose a file from the G2 server, by default, or from the client machine. The procedure allows you to create a file dialog for opening and saving files. The file dialog specification consists of a default pathname, a caption, the type of file dialog, a default file extension, and a filter. The filter allows the user to choose from a list of file extensions to filter the files that appear in the dialog.

g2-ui-choose-file
(*default-pathname*: text, *options*: structure, *window*: class g2-window)
-> *pathname*: text

Posts a dialog that allows the user to choose a path name on the file system of the G2 server, by default. The file dialog specifies a default path name and various options, including the dialog caption, dialog type, default file extension, and file type filter. This procedure returns the chosen path name as a text string, or "" if the dialog was cancelled.

options is a structure with this syntax:

structure
(caption: *text*,
save-file: *truth-value*,
extension: *text*,
filter: sequence (sequence *text*)
client-side: *truth-value*)

where:

- caption — Appears at the top of the dialog. The default value is "Open File". The default value when SAVE-FILE is true is "Save File".
- save-file — Determines whether to post a file dialog that saves or opens files, as follow:
  - true — Post a Save File dialog.
  - false — Post an Open File dialog, the default.
- extension — Provides the default file extension to use if none is supplied by the user. Specify the file extension without the "." (dot), for example, "kb". The default value is "".

- filter — A sequence of choices for the file type combo box, where each choice has the form (sequence (*label*, *wildcard*). The default value is:

  sequence(sequence("All Files (*.*)" "*.*"))

  which displays a single filter label in the combo box and allows files of any type, using the *.* wildcard:

  All Files (*.*)

  For example, to create a filter that allows `.kb` files and all files, the filter specification would look like this:

  ```
  sequence(sequence("KBs (*.kb)" "*.kb")
                sequence("All Files (*.*)" "*.*"))
  ```

- client-side — Determines whether to browse the file system of the G2 server (false), the default, or the machine associated with the given *window*, that is, the file system of the machine on which Telewindows is running (true).

If the choose-file-custom-buttons dialog type is supported on the window, then the structure supports two additional options for specifying the label of the OK and Cancel buttons: OK-BUTTON-LABEL and CANCEL-BUTTON-LABEL. Thus, the structure has this syntax:

structure
(caption: *text*,
save-file: *truth-value*,
extension: *text*,
filter: sequence (sequence *text*),
client-side: *truth-value*
ok-button-label: *text*,
cancel-button-label: *text*)

with default values:

ok-button-label— "Open or Save"

cancel-button-label — "Cancel"

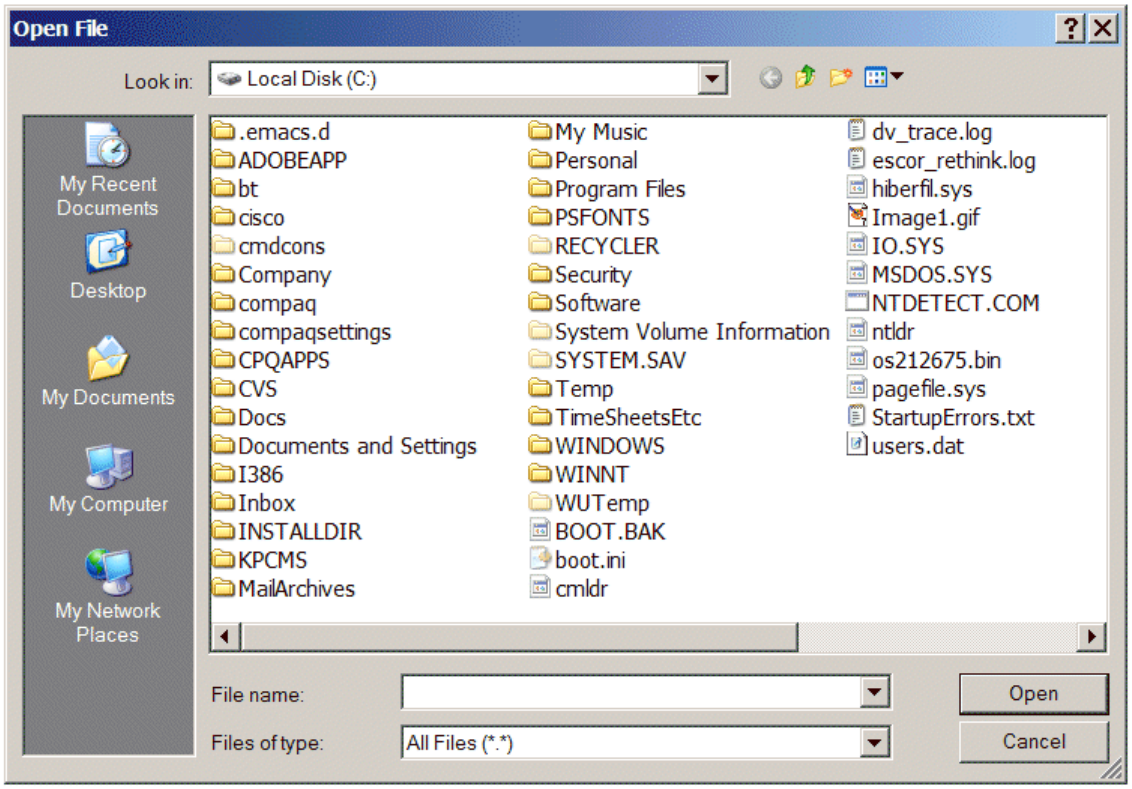
Here is a generic procedure that posts a file dialog and displays the chosen button in the Message Board:

TEST-CHOOSE-FILE

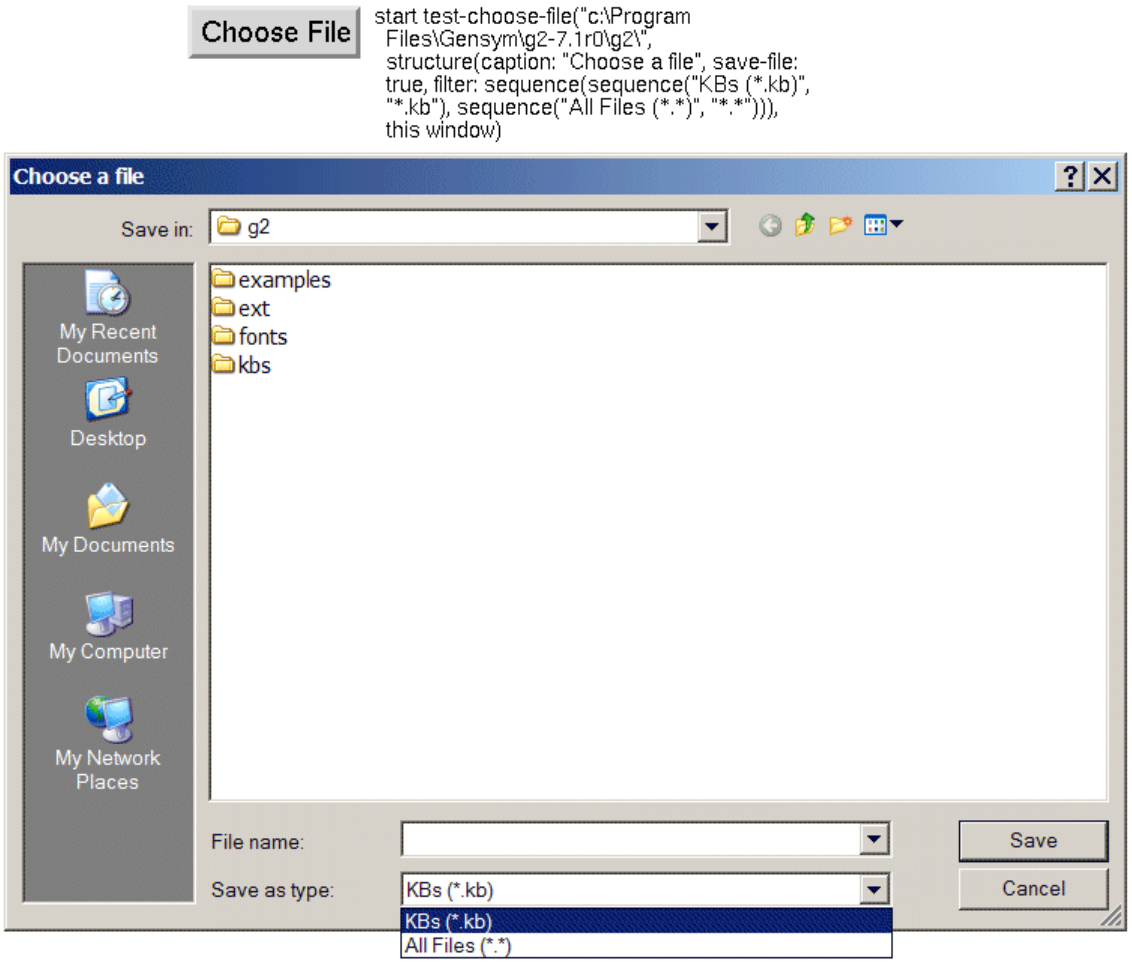
```
test-choose-file(default-pathname: text,
  options: structure, win: class g2-window)
result: text;
begin
 result = call g2-ui-choose-file(default-
   pathname, options, win);
 post "Result: [result]";
end
```

This example posts a file dialog that uses all the defaults, which creates an Open File dialog that allows files of any type:

Choose File start test-choose-file("c:\", structure(), this window)



This example posts a Save File dialog whose default path name is the default `g2` directory. The dialog specifies a caption and provides a filter for `.kb` files and files of any type.



When using the Windows file dialog in Telewindows to browse the file system of a G2 running on a different machine, the shortcut bar on the left of the dialog has shortcuts. The shortcuts depend on whether the server is running on a Windows or UNIX platform and on whether the G2 is local or remote, as follows:

- When connecting to a local G2 on Windows: My Recent Documents, Desktop, My Documents, My Computer, and My Network Places.
- When connecting to a remote G2 on any platform: The user's home directory, as defined by the `HOME` environment variable, and the directories in the module search path.

Directory Dialog

You can use the following system procedure to display a dialog for choosing a directory, as opposed to a file. For example, you would use this system procedure to allow users to choose a directory in which to store log files.

g2-ui-choose-directory
(*pathname*: text, *options*: structure, *win*: class g2-window)
-> *directory*: text

Displays a dialog in *win* for choosing a directory. Returns the pathname of the chosen directory or "" if the dialog is cancelled.

Pathname is the pathname of the initially selected directory in the dialog. Use "" to display the platform's default directory, which is generally the current directory.

Options is a structure with the following syntax:

structure
(caption: *text*,
ok-button-label: *text*,
cancel-button-label: *text*,
client-side: *truth-value*,
root: *text*)

where:

- caption — The text to display as a caption for the dialog. The default is "".
- ok-button-label — Alternative text to display on the OK button. The default is "OK".
- client-side — Whether to browse the client's file system. The default is false, which displays the server's file system in the dialog.
- cancel-button-label — Alternative text to display on the Cancel button. The default is "Cancel".
- root — Pathname of a directory to be considered the root directory, as a text. The user will only be allowed to select a subdirectory of this directory. Default value is "/" for G2 running on a UNIX machine or "X:\" for G2 running on a Windows machine, where "X" is the drive letter for the current default directory.

Dialog Information

This API procedure allows you to test whether certain types of dialogs are supported in a given window.

g2-ui-dialog-is-supported
(*type*: symbol, *window*: class g2-window)
-> *supported*: truth-value

Returns true if the given *type* of dialog is supported on *window*. The possible values for type are:

basic
message
confirmation
yes-no
yes-no-cancel
retry-cancel
query
notification
delay-notification
choose-file
choose-file-custom-buttons
print-workspace

A type of basic returns true only if all of the above dialogs are supported.

This procedure tests whether each symbol in the given array is a type of supported dialog and posts the result to the Message Board:

TEST-DIALOG-SUPPORT

```
test-dialog-support(win: class g2-window)
torf: truth-value;
s: symbol;
begin
  for s = each symbol in SA do
   torf = call g2-ui-dialog-is-supported(s,win);
   post "Supports [s]: [torf]";
   end;
end
```

```
message, confirmation, yes-no, query,
  choose-file, notification, yes-no-maybe
```

SA

Basic Dialogs

This API procedure posts a basic dialog, which consists of a message, a set of buttons for the specified type of dialog, an icon, and a caption. You can create message, confirmation, yes-no, yes-no-cancel, and retry-cancel dialogs.

g2-ui-post-basic-dialog
(*message*: text, *options*: structure, *window*: class g2-window)
-> *button*: symbol

Posts a basic dialog of some type, waits for the user to respond to the dialog, and returns a symbol naming the button chosen. If the dialog is cancelled, cancel is returned, regardless of whether or not the dialog has a cancel button.

options is a structure with this syntax:

structure
(type: *symbol*,
icon: *symbol*,
caption: *text*)

where:

- type — The type of dialog. The default value is message.

  This table lists the basic dialog types and their associated buttons:

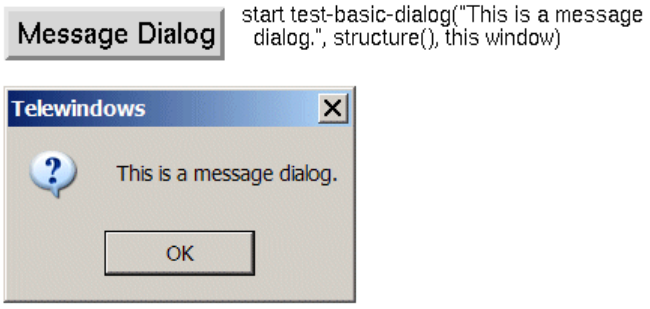
Type	Button
message	OK
confirmation	OK, CANCEL
yes-no	YES, NO
yes-no-cancel	YES, NO, CANCEL
retry-cancel	RETRY, CANCEL

- icon — The icon to display in the dialog. The default is question. The options are:
  - question — A question mark.
  - information — An "i" in a circle.
  - warning — A warning indicator.
  - error — An error indicator.
- caption — The name of the Telewindows window, which is Telewindows, by default.

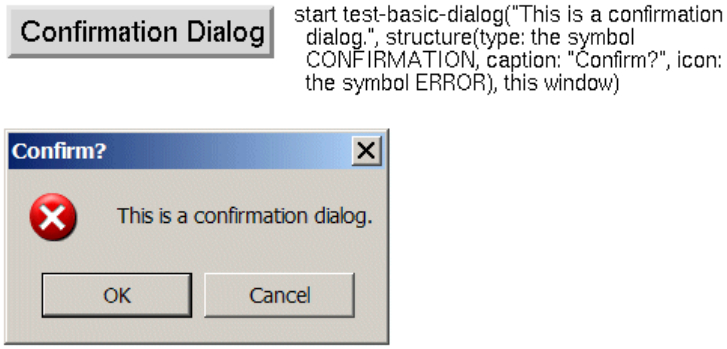
Here is a generic procedure that posts a basic dialog and displays the chosen button in the Message Board:

```
POST-BASIC-DIALOG
post-basic-dialog(msg: text, options:
  structure, win: class g2-window)
result: symbol;
begin
 result = call g2-ui-post-basic-dialog(msg,
   options, win);
 post "Result: [result]";
end
```

This example posts a simple message dialog that uses the default dialog type, caption, and icon, which results in a single button for accepting the dialog:



This example posts a confirmation dialog that specifies the dialog type, caption, and icon, which results in two buttons for accepting the dialog:



Query Dialog

This API procedure posts a query dialog, which consists of a message, a text field for the user to enter a value, a caption, and an icon. The dialog provides OK and Cancel buttons.

g2-ui-post-query-dialog
(*message*: text, *options*: structure, *window*:class g2-window)
-> *text*: text, *button*: symbol

Posts a query dialog, which has an icon, a message, a single-line text entry field, and OK and Cancel buttons, where *options* is a structure with this syntax:

structure
(caption: *symbol*,
icon: *symbol*)

where:

- caption — The name of the Telewindows window, which is Telewindows, by default.
- icon — The icon to display. The default is question. The options are:
  - question — A question mark.
  - information — An "i" in a circle.
  - warning — A warning indicator.
  - error — An error indicator.

This system procedure returns the entered text or "" if the dialog is cancelled. The resulting text string is limited to 1024 characters.

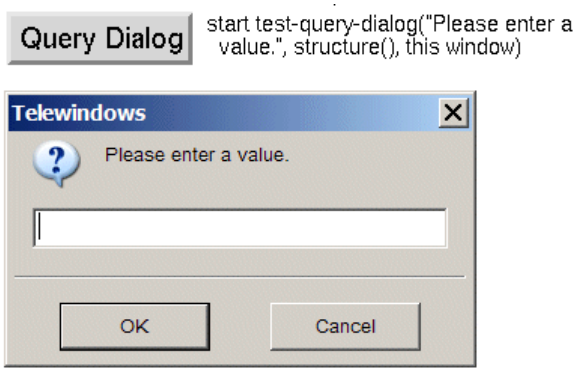
The *button* return value is one of these symbols: OK, CANCEL, or UNKNOWN. The return value is UNKNOWN only if the user entered no text and if you are running G2 Version 7.1 Rev. 0 or Rev. 1 with `sys-mod.kb` Version 7.1 Rev. 2.

Here is a generic procedure that posts a query dialog and displays the chosen button in the Message Board:

TEST-QUERY-DIALOG

```
test-query-dialog(msg: text, options: structure,
  win: class g2-window)
result: text;
button: symbol;
begin
 result, button = call g2-ui-post-query-
   dialog(msg, options, win);
 post "Result: [result] Button: [button]";
end
```

This example posts a query dialog that uses the default caption and icon, which results in two buttons for accepting the dialog:



Print Dialog

This API procedure posts a standard print dialog that allows the user to print a workspace directly to any printer on the client Telewindows.

g2-ui-print-workspace
(*workspace*: class kb-workspace, *options*: structure, *window*: class g2-window)

Prints *workspace* in one of several formats. By default, this procedure attempts to post a standard Windows print dialog on the specified *window*. The user can choose a printer and print the given *workspace*. If the user clicks the Print button, G2 generates a pair of logbook messages.

options is a structure with this syntax:

structure
(format: *symbol*,
 fmt: *symbol*,
 pathname: *text*,
 quality: *integer*,
 use-default-printer: *truth-value*)

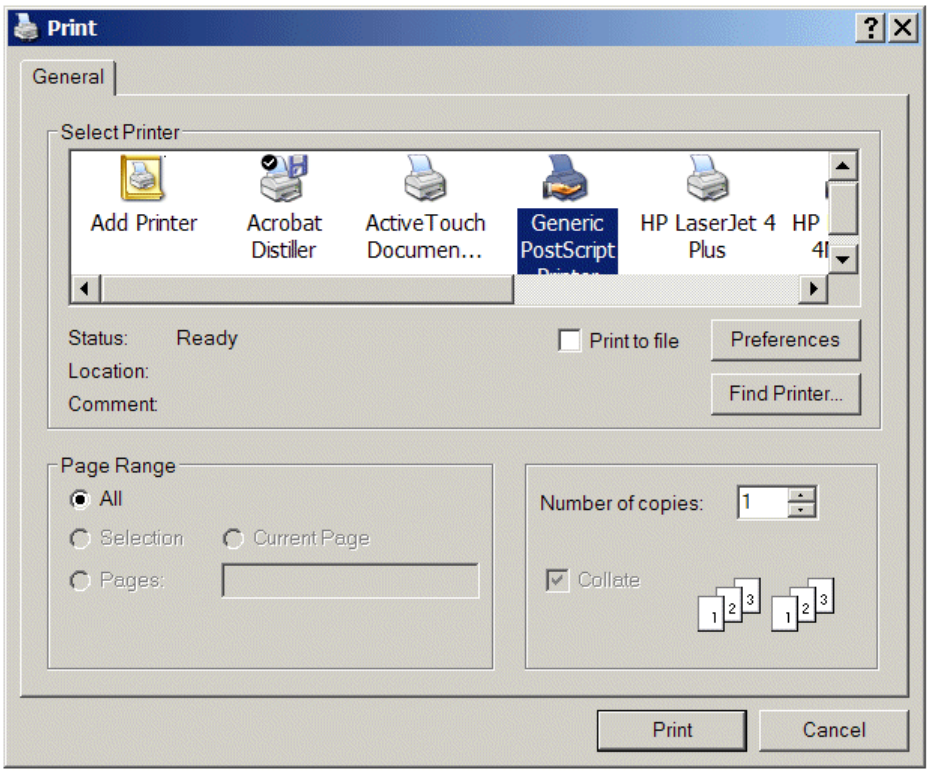
where:

- format — The print format, which is one of these symbols:
  - native — Displays a standard Windows dialog on the *window* for printing the workspace to a file on the client. This option is the default and only works if the client supports Windows dialogs.
  - postscript — Prints the workspace to a `.ps` file on the server.
  - jpeg — Prints to a `.jpg` file on the server.
- fmt — A synonym for format, which is a reserved word in G2.

- pathname — A pathname for the output file when format is postscript or jpeg. If a pathname is not specified, then the Printer Setup system table determines the default pathname, and the file type depends on the value of the format option, `.ps` for the postscript option and `.jpg` for the jpeg option.
- quality — The JPEG compression quality (0-100) when format is jpeg.
- use-default-printer — Whether to allow printing directly to the default printer from the Telewindows client, thereby avoiding the display of the print dialog. To print directly to the default printer from the client, specify use-default-printer as true. Note that some printers post their own dialog, even when the main print dialog has been skipped. Acrobat Distiller, for example, prompts for a destination pathname.

This example posts a Print File dialog:

```
Print Workspace   start g2-ui-print-workspace(this workspace,
                    structure(), this window)
```



Title Bar

g2-ui-set-window-title
(*title*: text, *window*: class g2-window)

Changes the G2 name in the title bar of the main G2 window. This procedure also sets the title bar text of the Telewindows main window.

WorkspaceView Control

A `WorkspaceView` ActiveX control is associated with a GSI interface through its connection with G2 ActiveXLink, as well as a G2 window through its connection with Telewindows. You use the following API procedures to get the G2 window and GSI interface associated with the `WorkspaceView` control.

For more information about the `WorkspaceView` control, see Chapter 7 "Using the WorkspaceView ActiveX Control" in the *Telewindows User's Guide*.

g2-ui-get-associated-g2-window
(*gsi-interface*: class gsi-interface)
-> *g2-window*: class g2-window

Returns the g2-window associated with the gsi-interface for the `WorkspaceView` control.

g2-ui-get-associated-gsi-interface
(*g2-window*: class g2-window)
-> *gsi-interface*: class gsi-interface

Returns the gsi-interface associated with the g2-window for the `WorkspaceView` control.

Show Workspace

g2-ui-show-workspace
(*workspace*: class kb-workspace, *options*: structure, *window*: class g2-window)

Shows a kb-workspace in a window, with the option of ensuring that the workspace is always visible in the given window. The *options* structure has one option:

structure
(ensure-visible: *truth-value*)

where:

- ensure-visible — When true, G2 ensures that at least some part of the workspace is visible within the given window. The default value is false.

Notification Dialogs

These API procedures post, update, and remove notification dialogs, which consist of a message, a caption, an icon, and font size, and delay notification dialogs, which consist of a message, a caption, and an animation object. Typically, you post a notification dialog, wait for some event to occur, update the dialog, then remove the dialog when some other event occurs. For delay notification dialogs, you can also start and stop the animation, and step through the animation one frame at a time.

g2-ui-post-notification
(*message*: text, *options*: structure, *window*: class g2-window)

Posts a small window containing an icon and a message in the center of the *window*. At most one notification window can be active per G2 window. No value is returned. *options* has this syntax:

structure
(caption: *text*,
icon: *symbol*,
font-size: *integer*)

where:

- caption — The caption to display in the dialog. The default value is "Notification".
- icon — The icon to display, as a symbol. The default is information. The options are:
  - question — A question mark.
  - information — An "i" in a circle.
  - warning — A warning indicator.
  - error — An error indicator.
- font-size — The font size for text, which is 14 pt.

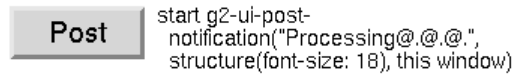
g2-ui-update-notification
(*message*: text, *options*: structure, *window*: class g2-window)

Changes the message in the active notification. *Options* are currently ignored.

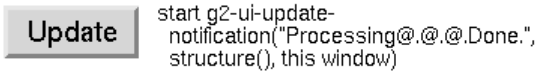
g2-ui-remove-notification
(*window*: class g2-window)

Removes the notification window from the window.

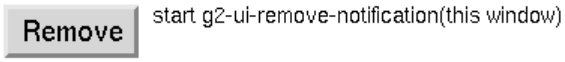
This example posts a notification dialog that uses the default caption and icon, but specifies a larger font size:



This example updates the notification dialog with new text:



This example cancels the notification dialog:



g2-ui-post-delay-notification
(*message*: text, *options*: structure, *window*: class g2-window)

Posts a small window containing a small clock with hands that move around the clock face and a message, in the upper-left corner of *window*. At most one delay notification window can be active per G2 window. No value is returned.

options is a structure with this syntax:

structure
(caption: *text*,
font-size: *integer*)

where:

- caption — The caption to display in the dialog. The default value is "Please wait...".
- font-size — The font size for text, which is 14 pt.

g2-ui-update-delay-notification
(*message*: text, *options*: structure, *window*: class g2-window)

Changes the message in the active delay notification window, if any. *Options* are currently ignored.

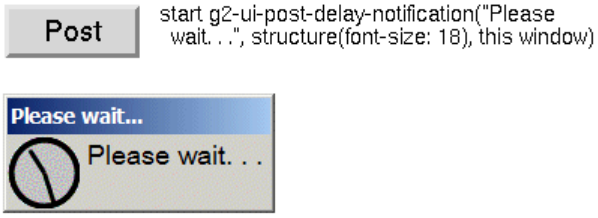
g2-ui-command-delay-notification
(*command*: symbol, *window*:class g2-window)

Starts, stops, or steps the animation forward one frame, according to the given *command* symbol. The options for *command* are: START, STOP, and STEP.

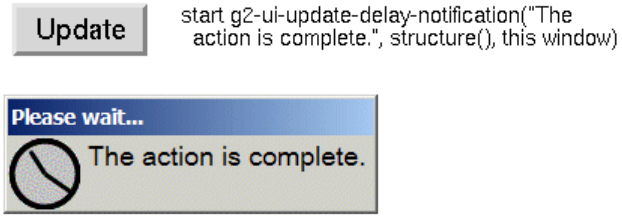
g2-ui-remove-delay-notification
(*window*: class g2-window)

Removes the delay notification dialog from the window.

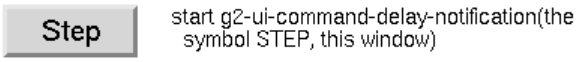
This example posts a delay notification dialog that uses the default caption and animation, but specifies a larger font size:



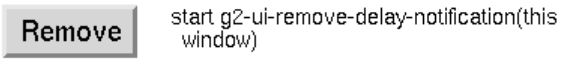
This example updates the notification dialog with new text:



This example steps through the animation one frame at a time:



This example cancels the notification dialog:



Editor Operations

Use the editor operations to configure the grammar prompts that appear in both the standard and classic G2 Text Editor.

g2-ui-insert-text-into-current-editor

g2-ui-get-text-from-current-editor

g2-ui-edit-text

g2-ui-launch-editor

g2-ui-close-current-editor

g2-manage-edit-session-on-window

g2-end-edit-session-in-window-if-any

g2-cancel-edit-session-in-window-if-any

g2-update-edit-session-in-window-if-any

g2-refresh-field-edit-from-message

g2-refresh-field-edit-with-text

g2-ui-insert-text-into-current-editor

Inserts *text* at the current cursor position in the current G2 Text Editor. Currently, no options are supported.

Synopsis

g2-ui-insert-text-into-current-editor
(*text*: text, *options*: structure, *window*: class g2-window)

g2-ui-get-text-from-current-editor

Get *text* from the current G2 Text Editor.

Synopsis

g2-ui-get-text-from-current-editor
    (*window*: class g2-window)
    -> *text*: text

g2-ui-edit-text

Launches the text editor on a given text string, waits for editing to finish, then returns the resulting text string and status.

Synopsis

g2-ui-edit-text
(*text*: text, *options*: structure, *window*: class g2-window)
-> *text*: text, *status*: symbol

Argument	Description
text	The text string to edit.
options	A structure with syntax as follows: structure (attribute: *symbol*, exclude: *sequence,* preferred-editor: classic | native, grammar: *symbol* | *structure*) where: • attribute is the attribute of the item to edit in the text editor. Items such as procedures, rules, messages, and free text, which provide the edit menu choice, do not require that you specify the attribute to edit. • exclude is a sequence of text strings or symbols to exclude as prompts in the text editor. Editor prompts that begin with an element of the exclude sequence do not appear. • preferred-editor overrides the value of the preferred-editor attribute in the Editor Parameters system table, which determines whether to use the classic or standard text editor in Telewindows. The default value for this attribute is classic. For details, see Editing Text in Using Telewindows in the *Telewindows User's Guide*. • grammar is either a symbol naming a class or a structure with the following syntax: structure (class: *symbol*, attribute: *symbol*, defining-class: *symbol*)
window	The window on which to display the editor, which can be a classic G2 or standard Telewindows window.

Return Value	Description
text	The resulting text string.
status	The symbol OK or the symbol CANCELLED.

g2-ui-launch-editor

Launches the text editor allowing for customization of the prompts that appear in the standard and classic G2 Text Editor.

Synopsis

g2-ui-launch-editor
(*item*: class item, *options*: structure, *window*: class g2-window)

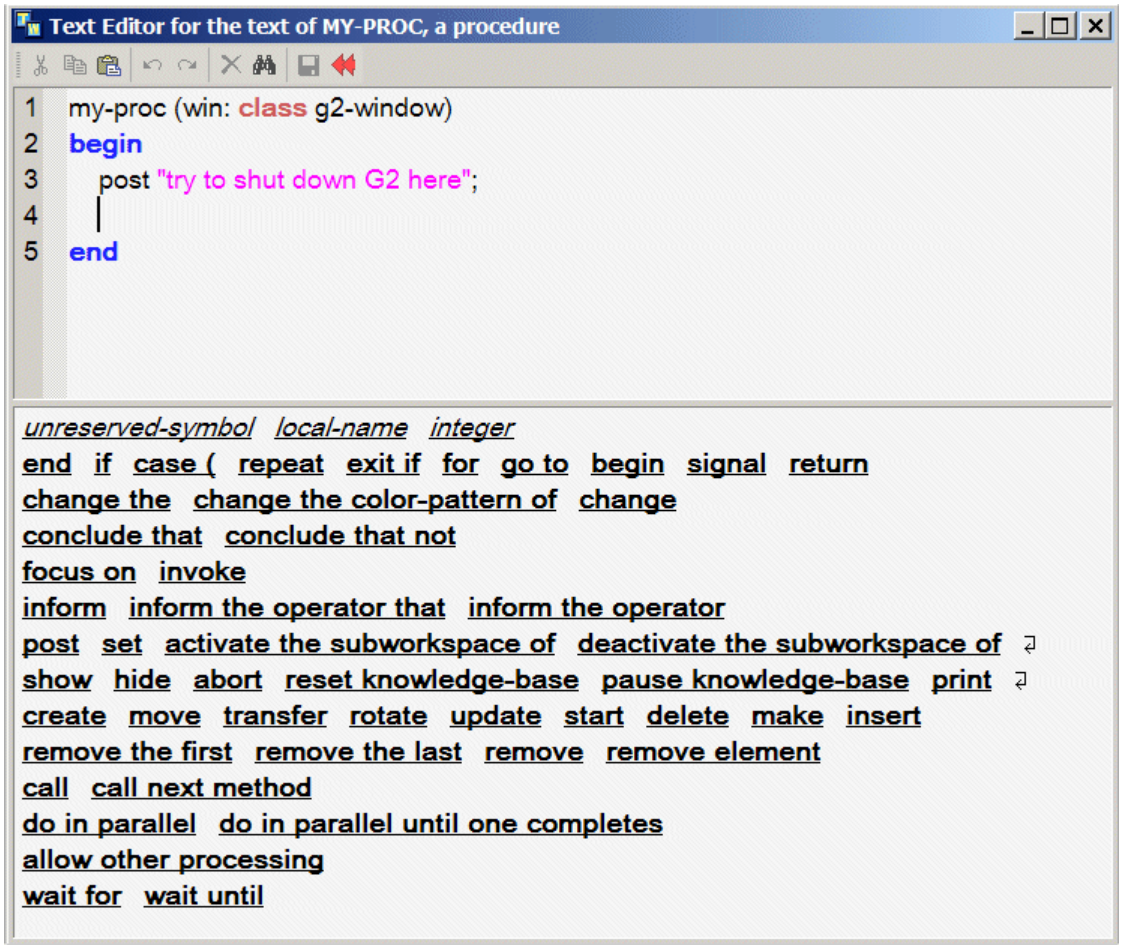
Argument	Description
item	The item to edit.
options	A structure with syntax as follows: structure (attribute: *symbol*, exclude: *sequence*, preferred-editor: classic | native, grammar: *symbol* | *structure* line: *integer* column: *integer*) where: • attribute is the attribute of the item to edit in the text editor. Items such as procedures, rules, messages, and free text, which provide the edit menu choice, do not require that you specify the attribute to edit. • exclude is a sequence of text strings or symbols to exclude as prompts in the text editor. Editor prompts that begin with an element of the exclude sequence do not appear. • preferred-editor overrides the value of the preferred-editor attribute in the Editor Parameters system table, which determines whether to use the classic or standard text editor in Telewindows. The default value for this attribute is classic. For details, see Editing Text in Using Telewindows in the *Telewindows User's Guide*. • grammar is either a symbol naming a class or a structure with the following syntax: structure (class: *symbol*, attribute: *symbol*, defining-class: *symbol*) • line and column can be used to specify the cursor position of the opened editor.
window	The window on which to display the editor, which can be a classic G2 or standard Telewindows window.

Example

This example launches the text editor for my-proc, excluding several editor prompts:

```
start g2-ui-launch-editor
    (my-proc, structure (exclude: sequence ("shut down g2", "halt", "collect data")),
    this window)
```

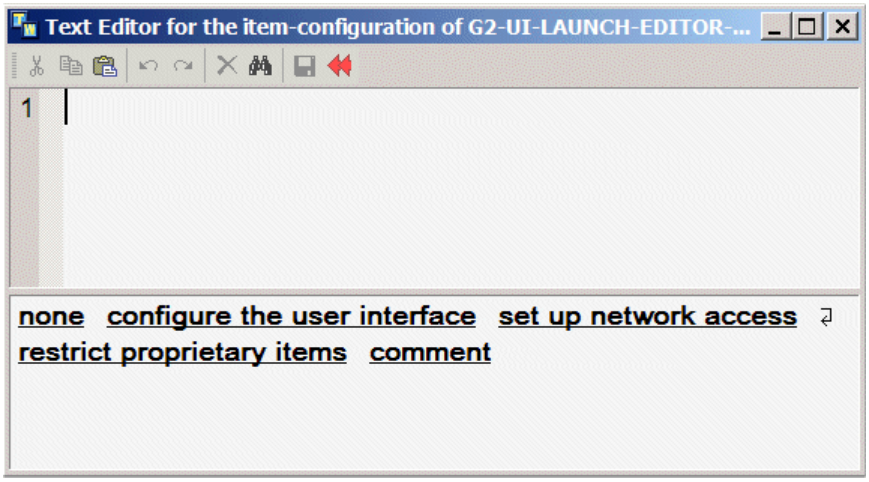
Here is the resulting text editor after placing the cursor after the post statement. Notice that shut down g2, halt, and collect data are not in the list.



This example launches the text editor for the item-configuration attribute of the current workspace, excluding the declare properties prompt:

```
start g2-ui-launch-editor
    (this workspace, structure (attribute: the symbol ITEM-CONFIGURATION,
    exclude: sequence ("declare properties")), this window)
```

Here is the resulting text editor. Notice that declare properties is not in the list of editor prompts.



g2-ui-close-current-editor

Closes the current active text editor with modifications saved or not.

Synopsis

g2-ui-close-current-editor
(*apply-changes*: truth-value, *window*: class g2-window)

Argument	Description
apply-changes	Apply changes when closing the editor.
window	The window on which to display the editor, which can be a classic G2 or standard Telewindows window.

g2-manage-edit-session-on-window

This is a system procedure available only through the UIL API. It launches the editor on message in window, just as if the user had used the mouse to invoke it. This is intended to be used in a situation where a message serves as a "field", e.g., for form-fill-in, dialogs, numeric entry, etc.

As is traditional with most "field editors", the TAB key forces the editing session to end, accepting whatever input (if possible, which it should always be for a message), and exits the editor. Thus, in this case the Tab cannot be used for its traditional purpose, namely to add horizontal whitespace to the text being edited.

The message must be on a workspace currently showing in window. The main reason is that it is not possible to edit in place unless the workspace on which you're editing is showing.

How the editor exits is flagged to the caller through the first return value, which may be one of the following symbols:

- cannot-enter-for-some-unknown-reason
- cannot-enter-because-restricted
- cannot-enter-because-not-showing
- cannot-enter-because-field-edit-in-progress
- abort-exit
- error-exit
- cancel-action
- tab-exit
- end-action
- end-exit
- cannot-enter-because-attribute-unknown

Presumably, the caller will use tab-exit as its cue to advance the user's input to the next "field". However, there is nothing "built-in" that enforces, or assists, in such behavior (e.g., no notion of "tab stobs").

Note that whether editing is permitted, as well as how the editor behaves (e.g., edit-in-place or not), can be modified by the user restrictions on message and on the mode window is in at the time of invocation.

This routine allows other processing. Deletions that may occur are dealt while editing are dealt with reasonably.

Synopsis

g2-manage-edit-session-on-window
(*message*: class message, *window*: class ui-client-item,
style: class uil-field-edit-style, *class-or-false*: item-or-value,
attribute-or-false: item-or-value,
defining-class-or-false: item-or-value)

Argument	Description
message	The message to edit.
window	The window on which to display the editor, which can be a classic G2 or standard Telewindows window.
style	The editing style.
class-or-false	The class information used to specify the grammar.
attribute-or-false	The attribute of class used to specify the grammar.
defining-class-or-false	The class of attribute used to specify the grammar.

g2-end-edit-session-in-window-if-any

If it finds a field edit in progress, it ends it.

Synopsis

g2-end-edit-session-in-window-if-any
(*window*: class ui-client-item)

Argument	Description
window	The window on which to cancel edit, which can be a classic G2 or standard Telewindows window.

g2-cancel-edit-session-in-window-if-any

If it finds a field edit in progress, it cancels it.

Synopsis

g2-cancel-edit-session-in-window-if-any
(*window*: class ui-client-item)

Argument	Description
window	The window on which to cancel edit, which can be a classic G2 or standard Telewindows window.

g2-update-edit-session-in-window-if-any

If it finds a field edit in progress, it updates it, returning true if successful.

Synopsis

g2-update-edit-session-in-window-if-any
(*window*: class ui-client-item)
-> *result*: truth-value

Argument	Description
window	The window on which to cancel edit, which can be a classic G2 or standard Telewindows window.

Return Value	Description
result	Returning true if successful.

g2-refresh-field-edit-from-message

If it finds a field edit in progress, it updates the edit buffer with the text of message.

Synopsis

g2-refresh-field-edit-from-message
(*message*: class message, *window*: class ui-client-item)

Argument	Description
message	The message for update.
window	The window on which to cancel edit, which can be a classic G2 or standard Telewindows window.

g2-refresh-field-edit-with-text

If it finds a field edit in progress, it updates the edit buffer with the text-value.

Synopsis

g2-refresh-field-edit-from-message
(*message*: class message, *window*: class ui-client-item,
text-value: text)

Argument	Description
message	The message for update.
window	The window on which to cancel edit, which can be a classic G2 or standard Telewindows window.
text-value	The target text for update.

Graphics-Oriented Operations

Use the graphics-oriented system procedures to direct G2 to reread an image-definition from a file, determine icon decaching management, and execute drawing and printing tasks.

g2-refresh-image-definition

Rereads an image-definition file from disk.

Synopsis

g2-refresh-image-definition
(*image*: class image-definition)

Argument	Description
image	The name of the image definition whose image file you want to refresh programmatically.

Description

This system procedure rereads the image definition file from disk, updating any icons that depend on it. For more information about using image definitions, see External Images in the *G2 Reference Manual*.

Example

A procedure that shows the workspace of an image, refreshes the image, and then hides the workspace, is:

```
show-and-refresh-image-on-workspace(image: class image-definition
    image-workspace: class kb-workspace)
begin
    show image-workspace at three-quarter scale at (250, -250) in the screen;
    call g2-refresh-image-definition(image);
    wait for 5 seconds;
    hide image-workspace
end
```

g2-set-icon-decaching-parameters

Specifies the size and decaching behavior of a class's icon image cache. This procedure is relevant only to classes whose icons change during KB execution.

Synopsis

g2-set-icon-decaching-parameters
(*class-1*: class class-definition, *maximum-renderings-for-class*: integer, *number-of-renderings-to-decache-at-a-time*: integer)

Attribute	Description
class-1	The class whose icon cache is configured by the call.
maximum-renderings-for-class	The maximum number of pixmaps to store in the class's icon cache. • Maximum: None in G2, but the operating system may set a limit. • Default: 200 • Minimum: 2
number-of-renderings-to-decache-at-a-time	The number of pixmaps to decache whenever the icon cache becomes full. • Maximum: (*maximum-renderings-for-class* - 1) • Default: 10 • Minimum: 1

The *number-of-renderings-to-decache-at-a-time* must be less than the *maximum-renderings-for-class*, or G2 signals an error and leaves the existing values unchanged.

Description

An icon that is visible on a workspace exists within G2 as a pixmap. All icons that have the same appearance use the same pixmap. Icons that have different appearances use different pixmaps. Every G2 class has an associated **icon cache** that holds every pixmap that G2 has created for any item of that class, up to the limit of the cache size. G2 uses this cache to improve performance: retrieving an existing pixmap from a cache is much faster than creating a new one.

When an icon first becomes visible, G2 creates a pixmap as specified by the class's icon description, caches the pixmap, and displays it on the workspace. When the user or the KB specifies a change to the icon's appearance, G2 first checks the icon

cache to see whether a pixmap having the specified appearance already exists. If one exists, G2 displays it. Otherwise, G2 creates, caches, and displays a new pixmap.

When an icon cache becomes full, G2 automatically decaches some of the least recently used pixmaps to make more space available. By default, an icon cache holds 200 pixmaps, and decaches them in blocks of 10. You can use g2-set-icon-decaching-parameters to change these defaults. Such changes are useful in two contexts:

- To avoid encountering operating system limitations on icon cache space.
- To improve performance when icon images are frequently or never reused.

Using g2-set-icon-decaching-parameters

The usual practice is to invoke g2-set-icon-decaching-parameters once in an initially rule, but you can invoke it at any time and any number of times. Reducing the size of an icon cache causes automatic decaching as needed to prevent overflow.

You must specify a value for both *maximum-renderings-for-class* and *number-of-renderings-to-decache-at-a-time* when you invoke the procedure. If you only want to change one of them, specify the default or the current value for the other.

Avoiding Operating System Limitations

Some operating systems restrict the space available to G2 for caching icon images. Attempting to exceed this limit aborts G2. The abort message includes a line such as:

```
Error: Unable to allocate M by N pixmap
```

G2 applications rarely reach an operating system limit on icon caching unless the limit is very restrictive and one or more classes generates very large and many icon pixmaps. As with other memory management considerations, thorough KB testing under maximum load is essential.

If your KB aborts due to lack of icon cache space, you can use g2-set-icon-decaching-parameters on the relevant class(es) to set a *maximum-renderings-for-class* that is low enough to avoid reaching the operating system limit. G2 then will never reach that limit, and will not abort.

Increasing Cache Size to Improve Performance

The more icon pixmaps that are cached when G2 needs them, the faster icon rendering becomes. Fastest performance results when all are cached. You can use *maximum-renderings-for-class* to increase icon cache size to improve performance, but be sure to run tests that actually use the allocated size, as described in the previous section. Unused cache space generates no overhead.

Tip The automatic decaching algorithm always maintains some freespace. If you know exactly how many pixmaps a class needs, and the icon cache can hold them all, specify a cache size that is one greater than the number of pixmaps. The unused capacity prevents unnecessary decaching to preserve freespace.

Avoiding Unnecessary Decaching

Decaching icon images incurs an overhead that is independent of the number of pixmaps to be decached, so each decaching should reclaim as much space as possible without causing pixmaps to be needlessly regenerated. Use *number-of-renderings-to-decache-at-a-time* to specify the number of pixmaps to decache when an icon cache becomes full.

The ideal *number-of-renderings-to-decache-at-a-time* depends on how a KB uses and changes icons, so no formula can be given. The default of 10 gives good results in most cases, but performance optimization tests can sometimes find better values.

Optimizing Performance When Icons Are Not Reused

When no icon pixmap is reused, icon caching just adds overhead, but G2 cannot know that and continues to maintain the icon cache. The resulting overhead is not large, but can be worth reducing in some cases.

- Setting *maximum-renderings-for-class* to lower values reduces memory use and cache search time.
- Setting *number-of-renderings-to-decache-at-a-time* to higher values reduces the number of decaching operations.

When icons are not reused and icon cache overhead must be minimized, the *number-of-renderings-to-decache-at-a-time* should be one less than the *maximum-renderings-for-class*. This pattern minimizes time spent decaching for any cache size.

The best value for *maximum-renderings-for-class* then depends on an whether time or space must be conserved. A larger cache uses more space while decaching less frequently. A smaller cache saves space but requires more frequent decaching.

g2-set-icon-bitmap-decaching-parameters

Sets the maximum number and maximum storage size for bitmap images.

Synopsis

g2-set-icon-bitmap-decaching-parameters
(*maximum-icon-bitmaps*: integer, *maximum-icon-bitmap-memory*: integer)

Argument	Description
maximum-icon-bitmaps	The maximum number of bitmaps G2 will be allowed to create for caching icons, per window. The default value is 1000.
maximum-icon-bitmap-memory	The maximum amount of storage G2 will be allowed to use for icon bitmap caches, per window, in bytes. The default is 2000000 (2MB).

Description

The values are on a per-window basis.

g2-work-on-drawing

Forces G2 to run drawing tasks upon demand.

Synopsis

g2-work-on-drawing
(*ws-or-boolean*: item-or-value)
-> *drawing-complete*: truth-value

Argument	Description
ws-or-boolean	The value you pass to the system procedure, which can be true, false, or a workspace item.

Return Value	Description
drawing-complete	A truth-value that returns true if drawing completes, or false if it does not.

Description

This system procedure directs G2 to run drawing tasks, regardless of other processing, returning true when complete. Use this procedure to complete drawing before continuing another task, or to test for complete drawing. Drawing occurs at the moment G2 calls the procedure.

The system procedure responds to the arguments you can provide:

Use this argument...	To...
false	Test whether drawing is complete. With this argument, the system procedure does not do any processing. It returns false if drawing is pending on any workspace, or true if drawing is complete for all workspaces.

Use this argument...	To...
true	Complete drawing tasks for the entire KB. The system procedure completes all pending drawing tasks for every workspace, returning true when it completes. No other processing occurs while the system procedure is executing.
kb-workspace	Run pending drawing tasks for that workspace for a short time. The procedure returns true if drawing completes, or false if the workspace drawing is incomplete.

g2-work-on-printing

Forces G2 to run printing upon demand.

Synopsis

g2-work-on-printing
(*ws-or-boolean*: item-or-value)
-> *printing-complete*: truth-value

Argument	Description
ws-or-boolean	The value you pass to the system procedure, which can be true, false, or a workspace item.

Return Value	Description
printing-complete	A truth-value that returns true if printing is complete, or false if it is not.

Description

This system procedure directs G2 to run a print task, regardless of other processing, returning true when complete. Use this procedure to complete printing tasks or to test for complete printing. Printing occurs at the moment the procedure is called.

The system procedure responds to the arguments you can provide:

Use this argument...	To...
false	Test whether printing is complete. With this argument, the system procedure does not complete any processing. It returns false if printing is pending on any workspace, or true if printing is complete for all workspaces.

Use this argument...	To...
true	Complete printing tasks for the entire KB. The system procedure completes all pending printing tasks for every workspace, returning true when it completes. No other processing occurs while the system procedure is executing.
kb-workspace	Run pending printing tasks for that workspace for a short time. The procedure returns true if printing completes for the workspace, or false if printing is incomplete.

HTML Views

G2 provides a Windows HTML view, which allows you to embed an Internet Explorer Web Browser window into Telewindows.

Note HTML views are only supported in Telewindows Next Generation (`twng.exe`).

For examples, see Using HTML Views in Windows Views, Panes, and UI Features in the *G2 Reference Manual*.

g2-ui-create-html-view
(*url*: text, *callback*: symbol, *options*: structure, *win*: class g2-window)
-> *handle*: integer

Creates an HTML view as an MDI child on the given window and returns an integer handle to the view, where *url* is the initial URL to display in the view. If the g2-window does not support HTML views, an error is signalled.

The *callback* is a procedure that gets called when the user closes the HTML view. Specify the callback as the symbol none to provide no callback. The syntax of the callback procedure is:

my-html-view-callback
(*event*: symbol, *win*: class g2-window, *control*: integer, *item*: value, *info*: structure, *user-data*: value)

Argument	Description
event	A symbol naming the event that invoked the callback. The only option is closed. The callback also responds to the event types described in View Events.
win	The window in which the HTML view appears.
control	The integer handle of the HTML view.
item	Currently, *item* is "none".

Argument	Description
info	A structure that describes the HTML view, as follows: structure (handle: *integer*, left: *integer*, top: *integer*, width: *integer*, height: *integer*, state: *symbol*, class: *symbol*, name: *symbol*) where: • handle is the integer handle of the HTML view. • left and top are integers that describe the location of the left and top edges when the view is closed. • width and height are integers that describe the width and height of the view when it is closed. • state is normal or minimized. • class is the G2 class for the HTML view. • name is not yet implemented.
user-data	For the closed event, the symbol none.

The *options* argument is a structure with this syntax:

structure
(left: *integer*,
top: *integer*,
width: *integer*,
height: *integer*,
container: *symbol-or-integer*,
neighbor: *value*,
dock: *symbol*)

where:

- left is the initial position of left side of the view window, in pixels.
- top is the initial position of top of the view window, in pixels.

- width is the initial width of view window, in pixels.
- height is the initial height of view window, in pixels.
- container is one of:
  - pane, as a symbol, which displays the view in a docked pane.
  - mdi-child, as a symbol, which displays the view in a floating pane. The default is mdi-child.
  - A dialog handle, as an integer, which displays the view in a dialog. The view is placed in the dialog, replacing the existing control, which must be a label. When container is a dialog handle, the neighbor attribute can be the control ID of another dialog against which to dock, in which case the dock attribute must be the symbol within.
  - The handle of a listbar-style shortcut bar, in which case the neighbor option is the number of the folder within the listbar into which to create the view.
- neighbor — Another pane against which to dock, as a view designation. See g2-ui-is-valid-window-handle for a description of a view designation. The default value is the overall frame.

  To specify that the view be placed in a tab pane within another pane, specify these options: dock: the symbol within, neighbor *h*, where *h* is the pane within which to place the view pane. When container is a dialog handle, the neighbor attribute can be the control ID of another dialog against which to dock.

- dock — The docking edge, which can be one of these symbols: left, top, right, bottom, float, or within. The default value is left.

g2-ui-manage-html-view
(*action*: symbol, *handle*: integer, *arg:* value, *win*: class g2-window)

Sends commands to an existing HTML view.

Argument	Description
action	The command name to execute. The following commands are supported: • goto navigates to the URL given in *arg*. • back navigates the previous web page. • forward navigates the next web page. • home navigates to the current user's home page. • stop stops fetching current page. • refresh reloads current page. • destroy destroys the HTML view.
handle	The handle to the HTML view as returned by g2-ui-create-html-view.
arg	An argument to the command. The argument is ignored for all commands except goto, where the argument is the desired URL, as a text string.
win	The g2-window on which the HTML view was created.

g2-ui-html-help
(*command*: symbol, *options*: structure, *window*: class g2-window)

Launches, manages, or destroys a Windows HTML Help window on the client for a given Windows help (`.chm`) file.

The options for *command* are:

- display-topic — Displays a topic given by the topic name or integer ID.
- display-contents — Displays the Table of Contents tab. The topic option in the structure has no effect.
- display-index — Displays the Index tab and scrolls to the specified index entry.

- display-popup — Displays a string in a popup window.
- destroy — Destroys all help windows.

The *options* structure has this syntax:

```
structure
(help-file-directory: text,
 help-file-name: text,
 topic: text | integer)
```

where:

- help-file-directory — The directory of the `.chm` file to display. The default is the directory of the G2 or TW executable.
- help-file-name — The filename of the `.chm` file to display. The default is "Master.chm".
- topic — The topic name or topic ID to display. The default is "". The topic name can be the name of a `.html` file included in the `.chm` file, a complete URL reference to a topic within the `.chm` file, or an integer that is a topic ID as defined in the Windows Help header file, for example:

```
#define kbs15_html 1127  /* The Paused Run-State   */
```

Input-Handling Operations

Use the input-handling system procedures to access input-event information and direct G2 to determine the state of an item.

g2-last-input-event

g2-last-input-event-info

g2-system-predicate

g2-last-input-event

Returns a symbol of the last keyboard or mouse event.

Synopsis

g2-last-input-event
(*win*: class g2-window)
-> *last-event*: symbol

Argument	Description
win	The window from which you wish to capture the last keyboard or mouse event.

Return Value	Description
last-event	The symbolic value of the last keyboard or mouse event in *win*, described next.

Description

This system procedure returns a symbol of the last event from the keyboard or mouse for any G2 window. The window of interest is specified by the first argument.

If the last user event was...	The procedure returns this value...
Typing a lower-case character	The letter as an upper-case character.
Typing an upper-case character	The symbol SHIFT + letter.
Pressing a function key	The function key that was pressed (such as F1 through F6, TAB, and RETURN).
Pressing the mouse down	The symbol MOUSE-DOWN.
Releasing the mouse	The symbol MOUSE-UP.
Special character (such as ctrl@+alt@+q)	CONTROL+ALT+Q

Note You can precede all input events with additional keys such as Ctrl, Alt, and Shift. The function that the key strokes normally perform still occurs. For example, Ctrl + f makes the workspace full-scale.

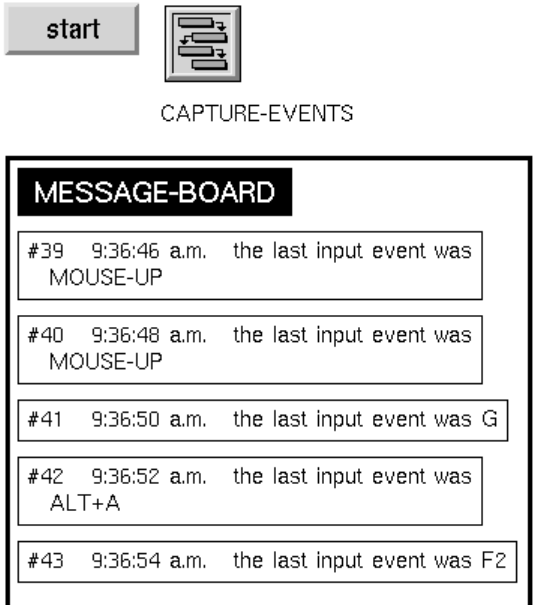
This procedure ignores mouse hover events. At startup, the event returned is UNKNOWN.

Example

An example indicating how to call the system procedure and post which user event last occurred, is:

```
capture-events( )
window: class g2-window;
event: symbol;
iteration: integer;
begin
    for iteration = 0 to 4 do
        for window = each g2-window
            do
                wait for 2 seconds;
                allow other processing;
                event = call g2-last-input-event(window);
                post "the last input event was [event]"
            end
    end
end
```

Click the start button.
Type a character, press a function key, enter a sequence (Alt +A), or click the mouse.
A message will display momentarily.



g2-last-input-event-info

Returns information regarding last keyboard or mouse event in any G2 window.

Synopsis

g2-last-input-event-info
(*win*: class g2-window)
-> *last-event*: symbol, *valid-item*: truth-value,
window-x: integer, *window-y*: integer, *timestamp*: integer,
workspace-x: integer, *workspace-y*: integer

Argument	Description
win	The window from which you wish to get information about the last keyboard or mouse event.

Return Value	Description
last-event	A symbol describing the last keyboard or mouse event in *win*. If the event type is a mouse event (mouse-up, mouse-down, mouse-motion, or mouse-select), the return value indicates which mouse-button initiated the activity. For a three-button mouse, the value can be left, middle, or right, and for a two-button mouse, left or right. If the event type is not a mouse event, returns the symbol none.
valid-item	A truth-value describing whether the event was over a valid item. When *valid-item* returns true, then the last two values returned are the coordinates of the event in the workspace of the item. If this is false, then the last two values will be 0.
window-x	The window x-position of the event as an integer.
window-y	The window y-position of the event as an integer.

Return Value	Description
timestamp	The timestamp as an integer, in milliseconds, which is the same clock as user mouse tracking.
workspace-x	An integer representing the workspace x-position of the event, if any, else 0.
workspace-y	An integer representing the workspace y-position of the event, if any, else 0.

Description

This system procedure returns all information regarding the last event from the keyboard or mouse for any G2 window. The window of interest is specified by the first argument.

This procedure ignores mouse hover events. At startup, the event returned is UNKNOWN.

g2-system-predicate

Tests any item against a specific system predicate.

Synopsis

g2-system-predicate
(*item*: item-or-value, *predicate*: symbol)
-> *predicate-true*: truth-value

Argument	Description
item	The item to test against a system predicate.
predicate	An item's system predicate that you want to test. You can test whether an item is permanent, transient, or showing. The system predicate showing indicates whether an item is visible on *any* G2 window, not a specific window.

Return Value	Description
predicate-true	A truth value that is true if the item has the predicate, or false if it does not.

Description

This system procedure lets you test an item against a system predicate you provide, returning true if the item has that predicate, or false if it does not.

After determining an item's predicate, you could change it or determine more information about it. For instance, you could make a transient item permanent, and use an item with a showing system predicate as a reference point for other items.

Example

This example shows how to check to see if an item is transient:

```
is-transient-truth-value =
    call g2-system-predicate(bridge34, the symbol transient)
```

Native Menu System (NMS)

Use the Native Menu System (NMS) API to create and manipulate menus in Telewindows.

For a description of the system procedures, see Native Menu System (NMS) API.

Parsing Operations

The G2 parser and validation API is a collection of system procedures that provide an interface to the G2 side of the standard text editor. These system procedures also exist as RPCs, which means you can create your own editor-like functionality in an external system, such as a bridge process.

The APIs allow you to edit G2 procedures, functions, rules, and attributes in external systems, validate the syntax before it is committed, and commit the changes back into G2. G2 maintains a copy of the string to edit, and the external editor has a copy of the string. The external editor provides information to G2 when the string changes and when the cursor changes. G2 returns prompting information to the external editor.

The API consists of these categories of procedures:

- Parsing and committing API procedures text allows you to parse an attribute of an item to create a parsing context, to commit changes to the parsing context, and to clear and delete the parsing context.
- Validating text API procedure validates specified text within a given context, which is the attribute and class. The validation API returns a complex structure that indicates parsing errors and current cursor location.
- Editing text API procedures adjust the current cursor position, insert characters, and delete characters.

Parsing and Committing Text

g2-simple-create-parsing-context
(*Item*: class item, *AttributeName*: symbol)
-> *ParsingContextHandle*: integer

Parses the *AttributeName* of *Item* and returns the parsing context as an integer handle. You refer to this handle in the following API procedures.

g2-compile-parse-result
(*ParsingContextHandle*: integer, *EditingForRuntimeChange*: truth-value)
-> *return*: structure

Compiles the text associated with the specified parsing context and commits the changes. The procedure compiles the text from the beginning to the current cursor position; it does not compile the entire text. Thus, to compile the entire text, you must adjust the cursor position to the end before calling this procedure. This allows you to save the current text and current cursor position without exiting the editor.

The *EditingForRuntimeChange* option indicates whether the change should permanently affect the KB (false) or whether it should revert to the original value on reload or restart (true).

The *return* structure has this syntax:

```
structure
(ok: truth-value,
description: text,
error-index: integer)
```

where:

- ok is true if the compile succeeds, false otherwise.
- description is an error message when ok is false, none otherwise.
- error-index is the cursor position of the error when ok is false, false otherwise. With the error "this is incomplete", the error-index is false even when the description is a string. To work around this problem, add a space in at the end of your text before compiling.

g2-clear-parsing-context
(*ParsingContextHandle*: integer)
-> *valid*: truth-value

Clears all the data associated with the specified parsing context. Use this procedure when the user has deleted everything in the text and you want to reset. The procedure returns false if *ParsingContextHandle* is invalid.

g2-delete-parsing-context
(*ParsingContextHandle*: integer)
-> *valid*: truth-value

Deletes the specified parsing context. Use this procedure the user has closed the external editor. The procedure returns false if *ParsingContextHandle* is invalid.

Validating Text

g2-validate-parsing-text
(*ClassName*: symbol, *AttributeName*: symbol, *TheText*: text)
-> *result*: structure

Parses the specified text, given the context of a *ClassName* and *AttributeName*. Use this procedure to validate procedure text or other expressions without changing the text.

Here are some examples.

To parse the text specified text of a procedure named my-proc, use this syntax. Note that my-proc does not need to exist as a procedure, and the text is not saved in an actual procedure during the operation.

```
g2-validate-parsing-text
(the symbol PROCEDURE, the symbol TEXT, "my-proc() begin end ")
```

To parse the text-attr attribute, which is defined to be a text, of an instance of my-class:

```
g2-validate-parsing-text
(the symbol MY-CLASS, the symbol TEXT-ATTR, "@"a string value@"")
```

To parse any expression, such as an expression in a readout table, you can use this syntax. Again, the readout table does not need to exist and is not created by this operation.

```
g2-validate-parsing-text
(the symbol READOUT-TABLE, the symbol EXPRESSION-TO-DISPLAY,
"8*9+17")
```

The return value is a structure with this syntax:

```
structure
(ok: boolean,
endable-p: boolean
description: {text | false},
error-index: {integer | false},
longest-common-completion: {text | false}
completion-choices: {sequence-of-strings},
category-choices: {sequence-of-strings},|
longest-common-completion: {text | false}
token-complete-p: boolean,
token-data: sequence)
```

- ok is true when the text up to the cursor position contains valid syntax, false otherwise.
- description is the text of the error message if ok is false.
- error-index is the location of the syntax error if ok is false.
- completion-choices contains the language prompts that are appropriate for the text, as a sequence of text strings, or an empty sequence.
- category-choices contains the categories that are appropriate for the text, as a sequence of strings, for example, "any integer", or an empty sequence.
- longest-common-completion contains the initial substring characters in common from completion-choices, if any.

Note The endable-p, token-complete-p, and token-data keywords of the structure are used for compatibility with Telewindows2 Toolkit only, and the token data are based on a zero-based index.

Editing Text

g2-adjust-cursor-position
(*ParsingContextHandle*: integer, *NewPosition*: integer)
-> *result*: structure

Adjusts the cursor position to *NewPosition*, which is a one-based index into the text. To move the cursor directly to the end of the text, pass a negative index. Usually the editor knows how long the string is supposed to be; however, it may be more convenient to go directly to the end of the text. This procedure throws an error if the index given is positive and too large.

See g2-validate-parsing-text for a description of the return value.

g2-parsing-context-delete-characters
(*ParsingContextHandle*: integer, *Position*: integer,
NumberOfCharacters: integer)
-> *result*: structure

Deletes the specified *NumberOfCharacters* from the text associated with the parsing context, beginning at *Position*, which is a one-based index into the text. The cursor shifts to *Position*, which the return structure reflects. This procedure throws an error if the index is invalid.

See g2-validate-parsing-text for a description of the return value.

g2-parsing-context-insert-characters
(*ParsingContextHandle*: integer, *Position*: integer, *TextToAdd*: text)
-> *result*: structure

Inserts *TextToAdd* into the text associated with the parsing context, beginning at *Position*, which is one-based index into the text. The cursor moves to the end of the addition. This procedure throws an error if the index is invalid, or if the insertion results in a string longer than the maximum size of a G2 string.

See g2-validate-parsing-text for a description of the return value.

g2-parsing-context-get-text
(*handle* : integer)
-> *text*: text

Returns the text string of the procedure or other item being edited by the specified parsing context *handle*. This procedure is useful for verifying that the remote side and the G2 side agree on what is being edited.

Property Grid Views

g2-ui-create-property-grid

g2-ui-create-property-grid
(*title*: text, *callback*: symbol, *options*: structure, *win*: class g2-window)
-> *handle*: integer

Creates a property grid view with the given *title* on *win*, returning an integer *handle*, or signalling an error if the window does not support property grids. The *callback* is a procedure to call when various events occur (see Callback). The *options* structure has these attributes:

- contents: *value* — See Contents below.
- container is one of:
  - pane, as a symbol, which displays the view in a docked pane. The default is pane.
  - mdi-child, as a symbol, which displays the view in a floating pane.
  - A dialog handle, as an integer, which displays the view in a dialog. The view is placed in the dialog, replacing the existing control, which must be a label. When container is a dialog handle, the neighbor attribute can be the control ID of another dialog against which to dock, in which case the dock attribute must be the symbol within.
  - The handle of a listbar-style shortcut bar, in which case the neighbor option is the number of the folder within the listbar into which to create the view.
- user-data: *item-or-value* — Arbitrary data, supplied to callback.
- show-help: *boolean* — Whether to show the help pane. Default is true.
- show-toolbar: *boolean* — Whether to show the toolbar. Default is false.
- view-divider: *float* — Fraction of width to devote to captions (0.0 - 1.0).
- help-height: *integer* — In pixels. Default is 56.
- foreground-color: *symbol* — A color symbol.
- background-color: *symbol* — A color symbol.
- line-color: *symbol* — A color symbol.
- category-foreground-color: *symbol* — A color symbol.
- read-only-foreground-color: *symbol* — A color symbol.
- help-foreground-color: *symbol* — A color symbol.

- help-background-color: *symbol* — A color symbol.
- sort: *symbol* — One of: none, alphabetical, or categorized. Default is categorized.

Contents

The contents of a property grid is expressed as a tree of nodes, where each node is either a category or a property:

- contents — *node* | sequence (*node*, ...)
- node — *category* | *property*
- category — structure (category: *category*, children: *children*, options: *options*)
- property — structure (property: *property*, type: *type*, current-value: *value*, category: *category*, options: *options*) | *symbol* | *text*
- children — sequence (*node*, ...)

See Properties and Categories for a description of these arguments.

Properties

The options for *property* are:

- property: *value* — Property name as a symbol or text. Required.
- label: *text* — Caption for property. Default is derived from name.
- type: *value* — See Types below. Default is based on current-value; otherwise text.
- category: *value* — See Categories below.
- current-value: *value* — Default is based on type; otherwise "".
- possible-values: *sequence* — A sequence of text strings. Creates a combo-box for editing the value.
- description: *text* — Default is "".
- value-format: *symbol* — dd.ddd symbol for floats, short-date (default) or long-date for dates.
- icon: *item-or-value* — Icon displayed beside value, used for text types only.
- readonly: *truth-value* — Default is false.
- ellipsis: *truth-value* — Whether to add an ellipsis button. Default is false.
- edit-in-place: *truth-value* — Whether to allow editing in-place. Default is true.
- select: *truth-value* — Whether item is selected. Default is false.
- user-data: *value* — Arbitrary value, supplied to callback.

Categories

The options for *category* are:

- category: *value* — Category name as a *symbol* or *text*. Required.
- label: *text* — Caption of the category. Default is derived from the name.
- expand: *truth-value* — Whether to initially expand category. Default is false.
- children: *sequence* — Default is sequence().

Categories can be created implicitly or explicitly:

- Implicitly: If a property node has a category option, then a category of that name is created if one does not already exist. If a property node has no category option, and is not in the children list of some category, then the category "Misc" is assumed.
- Explicitly: If a node has the category option but not the property option, then the named category is created, making use of the other options in the node. The members of the category can be listed explicitly in the children option.

Categories may only contain properties; nested categories are currently not supported.

Types

The following built-in types are currently available:

- category — A category.
- text — A text string with optional icon. Can be a combo-box.
- integer — An integer.
- boolean — A boolean truth-value, true or false.
- color — A G2 color, as a symbol.
- date — A date, expressed as either a G2 time, for example, the current real time, or a structure of the following form, where each attribute defaults to today's value:

  structure(month: *integer*, date: *integer*, year: *integer*)

- float — A floating point value.

Callback

The syntax for the *callback* argument is:

callback
(*event*: symbol, *window*: class g2-window, *propgrid*: integer, *name*: value, *info*: structure, *user-data*: item-or-value)

where:

- *event* — See Events below.
- *window* — The G2 window.
- *propgrid* — Handle of property grid.
- *name* — Name of relevant node, if any.
- *info* — Additional info: current-value, name, type, mouse x/y, etc.
- *userdata* — The user-data supplied for property grid, if any.

Events

The options for *event* specific to a property-grid are:

- select — Node was selected.
- edit — Ellipsis button was pressed on node.
- edited — Node value was edited by user locally in the client.
- click — Node was left-clicked.
- right-click — Node was right-clicked.
- node-expanded — Node with children was expanded.
- node-collapsed — Node with children was collapsed.

g2-ui-manage-property-grid

g2-ui-manage-property-grid
(*action*: symbol, *handle*: value, *arg*: value, *win*: class g2-window)

Perform an action on a property grid, possibly using the argument *arg*. Supported actions and the expected type for the *arg* are:

- clear — Clears the property grid (*arg* is ignored).
- destroy — Destroy the property grid (*arg* is ignored).
- populate — Populates the property grid, where *arg* is the complete new contents of the property grid, in the same format as the contents option of g2-ui-create-property-grid.
- sort — One of none, alphabetical, or categorized, where *arg* is a symbol.

- reset-colors — Reset all colors to their default values (*arg* is ignored).
- background-color — Set the background-color, where *arg* is a symbol.
- foreground-color — Set the foreground-color, where *arg* is a symbol.
- line-color — Set the line-color, where *arg* is a symbol.
- category-foreground-color — Set the category-foreground-color, where *arg* is a symbol.
- readonly-foreground-color — Set the readonly-foreground-color, where *arg* is a symbol.
- help-background-color — Set the help-background-color, where *arg* is a symbol.
- help-foreground-color — Set the help-foreground-color, where *arg* is a symbol.
- help-height — Height of help pane in pixels, where *arg* is an integer.
- show-help — Show or hide the help pane, where *arg* is a truth-value.
- show-toolbar — Show or hide the toolbar, where *arg* is a truth-value.

g2-ui-modify-node

g2-ui-modify-node
(*handle*: value, *node*: item-or-value, *changes*: structure,
window: class g2-window)

Modifies various attributes of the designated node in a property grid. *Node* can be the name, the label, or the integer handle of a node in the grid. The *changes* structure contains new values for any of the node's attributes, for example, current-value.

Overloaded System Procedures

The following system procedures, which work on tree views, also work on property grids:

g2-ui-select-node

g2-ui-expand-node

g2-ui-collapse-node

g2-ui-modify-node

g2-ui-insert-node

g2-ui-delete-node

For details, see Tree Views.

Reflection, Rotation, Size, and Layering Operations

Use the following system procedures to change the appearance and visibility of item icons.

g2-change-size-of-item-per-area

g2-combine-reflection-and-rotation

g2-drop-item-behind

g2-drop-item-to-bottom

g2-get-item-layer-position

g2-get-reflection-and-rotation

g2-lift-item-in-front-of

g2-lift-item-to-top

g2-move-from-area-of-workspace

g2-reflect-item-horizontally

g2-reflect-item-vertically

g2-restore-item-to-normal-size

g2-set-reflection-and-rotation

g2-change-size-of-item-per-area

Changes the size of an item to an area you specify.

Synopsis

g2-change-size-of-item-per-area
(*item*: class item, *left-edge*: integer, *top-edge*: integer,
right-edge: integer, *bottom-edge*: integer, *mode*: symbol)

Argument	Description
item	The item whose size you wish to change.
left-edge	The new left edge of the item.
top-edge	The new top edge of the item. This value must be greater than that for the bottom-edge.
right-edge	The new right edge of the item. This value must be greater than that for the left-edge.
bottom-edge	The new bottom edge of the item.
mode	Reserved for later use. Use the symbol default for this argument at this time.

Description

This system procedure changes the size of any item by providing the outer edges in workspace units as arguments to the procedure. It is a programmatic equivalent to the change-size item menu option. This procedure does not return a value; it changes the item size and returns. You cannot use this procedure to extend an item beyond its maximum size limit.

You cannot use this procedure to change the size of any item that does not include a change-size item menu option, such as dial, meter, and freeform-table items.

Example

To change the size of an item using this procedure requires that you first know the size and position of the item upon its workspace. You can get this information using the animation expressions, item-x-position, item-y-position, item-height, and item-width. Once you have those values, use them to calculate the left-edge,

top-edge, right-edge, and bottom-edge arguments of the system procedure, as in this example:

```
double-up(itm: class item)
left, right, top, bottom, new-height, new-width, new-x, new-y: integer;
begin
    { Double the current width and height. }
    new-height = the item-height of itm * 2;
    new-width = the item-width of itm * 2;
    new-x = round(new-width / 2);
    new-y = round(new-height / 2);

    { The new positions are based on the current x and y
        less half of the new size. }
    left = the item-x-position of itm - new-x;
    right = the item-x-position of itm + new-x;
    top = the item-y-position of itm + new-y;
    bottom = the item-y-position of itm - new-y;
    call g2-change-size-of-item-per-area(itm, left, top, right, bottom,
        the symbol default)
end
```

Note This procedure is functionally identical to the change size menu option. By specifying the target coordinates in a particular way, you could use it to move the item to another workspace position. While this is possible, we do *not* recommend that you do so. To move an item, use the g2-move-from-area-of-workspace system procedure.

The maximum size of an icon is 16 times the size given by its icon definition. For user-defined classes, you can obtain the size of an icon programmatically using an expression such as:

```
the width of the icon-description of my-class-definition
the height of the icon-description of my-class-definition
```

where my-class-definition is a class definition item.

g2-combine-reflection-and-rotation

Lets you combine a current reflection and rotation status of an item icon with a new value to obtain a certain result.

Synopsis

g2-combine-reflection-and-rotation
(*reflection-and-rotation-1*: symbol, *reflection-and-rotation-2*: symbol)
-> *combined-status*: symbol

Argument	Description
reflection-and-rotation-1	The first reflection-and-rotation status you wish to set on the item.
reflection-and-rotation-2	The second reflection-and-rotation status you wish to set on the item.

Return Value	Description
combined-status	A symbol of the combined reflection and rotation status.

Description

This system procedure accepts two symbolic reflection-and-rotation values as arguments and groups them to return a combined status.

The purpose of this procedure is to cumulate reflection and rotation in an item. Unlike the g2-set-reflection-and-rotation system procedure, which sets an item only from its normal position, you can use this procedure to essentially add to an item's current reflection and rotation to result in a combined reflection and rotation status.

Return Values

Here are the results of combining two reflection and rotation status symbols. When passing status symbols to the system procedure, enter the arguments in either order; the result is the same.

Combining this status...	With this status...	Returns this status...
reflected	normal	reflected
reflected	reflected	normal
reflected-clockwise-90	reflected	clockwise-270
reflected-clockwise-90	reflected-clockwise-90	normal
reflected-clockwise-180	reflected-clockwise-180	normal

Technique

To use this system procedure:

1. Use the g2-get-reflection-and-rotation system procedure to get an item's current reflection and rotation status. Alternatively, you can use the g2-reflect-item-horizontally or g2-reflect-item-vertically to get a particular status.
2. Use that value as the first argument of this system procedure.
3. For the second argument, use the status you want to add to the first.
4. Pass the return value of g2-combine-reflection-and-rotation to the g2-set-reflection-and-rotation system procedure to give the desired result.

Example

A procedure that uses these steps to get the current reflection and rotation of an object, and combine it with a reflected status, is:

```
combine-rotation (obj: class object) = (symbol)
current-status, combined-status, new-status: symbol;
begin
    current-status = call g2-get-reflection-and-rotation(obj);
    combined-status = g2-combine-reflection-and-rotation(current-status,
        the symbol reflected);
    new-status = call g2-set-reflection-and-rotation(obj, combined-status);
    return new-status
end
```

g2-drop-item-behind

Drops one item behind another.

Synopsis

g2-drop-item-behind
(*item-1*: class item, *item-2*: class item)

Argument	Description
item-1	The item you wish to drop to the bottom of the current stacking order on the workspace.
item-2	The item you wish to position on top of the first item.

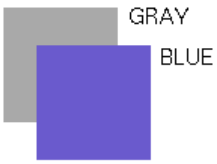
Description

This system procedure changes the stacking order of two items, *item-1* and *item-2*, on a workspace by changing the layer numbers. If the icons overlap, you can see that G2 places *item-1* beneath *item-2*.

Example

The next example places one item in front of or behind another, based on its item layer number.

The procedure tests to see which square is on top of the other, and adjusts the position accordingly, calling either g2-lift-item-in-front-of or g2-drop-item-behind.




```
lift-back-front(square1: class square, square2: class square)
square1-position, square2-position: integer
begin
    square1-position = call g2-get-item-layer-position(square1);
    square2-position = call g2-get-item-layer-position(square2);
    if square1-position < square2-position
        then call g2-drop-item-behind(square1, square2)
        else call g2-lift-item-in-front-of(square1, square2)
end
```

g2-drop-item-to-bottom

Drops an item to the bottom of the current stack of items.

Synopsis

g2-drop-item-to-bottom
(*item*: class item)

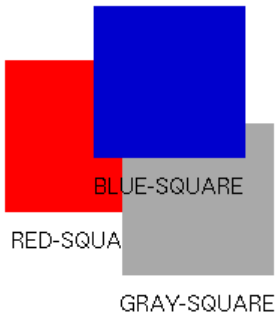
Argument	Description
item	The item you wish to drop to the bottom of the current stacking order on the workspace.

Description

This system procedure places the item you pass to it on the bottom of the stacking order on its workspace.

Examples

The next example shows a workspace with three square items. The procedure lifts blue-square to the top and, in turn, drops the other two squares to the bottom.




```
position-square-at-top(blue-square, ws: class-kb-workspace)
square: class square;
begin
    for square = each square upon ws
        do
            if square is the same object as blue-square
                then call g2-life-item-to-top(square)
                else call g2-drop-item-to-bottom(square)
        end
end
```

g2-get-item-layer-position

Returns an item's layer position.

Synopsis

g2-get-item-layer-position
(*item*: class item)
-> *layer-position*: integer

Argument	Description
item	The item whose layering position you wish to obtain. G2 returns an integer index indicating the position.

Return Value	Description
layer-position	An integer of the item layer position.

Description

Each item that resides upon a workspace has a layer number, indicating its position in a stack of items. As G2 moves items, adds to or deletes items from the workspace, it adjusts the layer numbers.

This system procedure gets the current layer number of an item, returning an integer index indicating the stacking order of the item on the workspace. The item with the lowest layer number is at the top of the stack of items. For instance, if three items have the layer numbers 5, 3, and 0, the item whose layer is 0 is on top of the stack of items.

For more information about item layering upon a workspace, see G2 Items in the *G2 Reference Manual*.

Example

A workspace has items with class designations that are unique on the workspace. The procedure iterates over every item upon the workspace, obtains the layer number, and displays the class and layer position on the message board.

```
display-layer-positions(ws: class kb-workspace)
layer-number: integer;
I: item;
begin
     for I = each item upon ws
          do
               layer-number = call g2-get-item-layer-position(I);
               post "The item layer position of the [the class of I] upon the
                    workspace is [layer-number]"
          end
end
```

g2-get-reflection-and-rotation

Returns a symbol of an item's current reflection and rotation status.

Synopsis

g2-get-reflection-and-rotation
(*item*: class item)
-> *reflection-rotation*: symbol

Argument	Description
item	The item whose reflection and rotation status you wish to obtain.

Return Value	Description
reflection-rotation	A symbol of the current reflection and rotation status, described next.

Description

This system procedure returns a symbol, indicating the current reflection and rotation status of an item. The return values cover all of the possible combinations of reflection and rotation for an item.

Reflection or rotation is applicable only to objects (items of the object class). For items that do not support reflection or rotation, such as workspaces and text-boxes, the system procedure always returns normal.

The g2-get-reflection-and-rotation system procedure returns one of the following symbols; the g2-set-reflection-and-rotation procedure accepts one of the symbols as its reflection-and-rotation argument:

If the value is...	Then the status of the item is...
normal	Neither rotated, nor reflected.
clockwise-90	Rotated by 90 degrees.
clockwise-180	Rotated by 180 degrees.
clockwise-270	Rotated by 270 degrees.
reflected	Reflected from left to right.

If the value is...	Then the status of the item is...
reflected-clockwise-90	Reflected left to right, and then rotated by 90 degrees.
reflected-clockwise-180	Reflected left to right, and then rotated by 180 degrees, which is the equivalent of being reflected upside down.
reflected-clockwise-270	Reflected left to right, and then rotated by 270 degrees.

g2-lift-item-in-front-of

Lifts one item in front of another.

Synopsis

g2-lift-item-in-front-of
(*item-1*: class item, *item-2*: class item)

Argument	Description
item-1	The item whose layer number you wish to make less than the other item such that, if the two items overlap, this item will be on top of the other.
item-2	The item whose layer number you wish to ensure is greater than the other item, and whose stacking order is below *item-1*.

Description

This system procedure changes the stacking order of two items on a workspace, *item-1* and *item-2*, by changing the layer numbers. If the icons overlap, you can see that G2 places *item-1* on top of *item-2*.

g2-lift-item-to-top

Lifts an item to the top of the current stack of items.

Synopsis

g2-lift-item-to-top
(*item*: class item)

Argument	Description
item	The item you wish to lift to the top of the current stacking order on the workspace.

Description

This system procedure raises the item you pass to it to the top of the stacking order on its workspace.

g2-move-from-area-of-workspace

Moves an area of items from one location to another.

Synopsis

g2-move-from-area-of-workspace
(*wksp*: class kb-workspace,
left-edge: integer, *top-edge*: integer, *right-edge*: integer,
bottom-edge: integer, *delta-x*: integer, *delta-y*: integer)

Argument	Description
wksp	The workspace containing the items you want to move as a group.
left-edge	The x coordinate of the left edge of the items to move.
top-edge	The y coordinate of the top-edge of the items to move.
right-edge	The x coordinate of the right edge of the items to move.
bottom-edge	The y coordinate of the bottom-edge of the items to move.
delta-x	The number of workspace units of the delta between where the items are currently and where they are moving along the x axis.
delta-y	The number of workspace units of the delta between where the items are currently and where they are moving along the y axis.

Description

Moves an area of items from one location on a workspace to another, providing a programmatic equivalent to the workspace Operate On Area menu option that lets you move one or more items. This system procedure does not return a value; it returns after moving the items.

g2-reflect-item-horizontally

Reflects an item horizontally and returns the current reflection status as a symbol.

Synopsis

g2-reflect-item-horizonally
(*item*: class item)
-> *current-reflection*: symbol

Argument	Description
item	The item you wish to reflect horizontally.

Return Value	Description
current-reflection	A symbol of the current reflection status of *item*.

Description

This system procedure reflects an item horizontally, returning the current reflection-and-rotation status of the item.

g2-reflect-item-vertically

Reflects an item vertically and returns the current reflection status as a symbol.

Synopsis

g2-reflect-item-vertically
(*item*: class item)
-> *current-reflection*: symbol

Argument	Description
item	The item you wish to reflect vertically.

Return Value	Description
current-reflection	A symbol of the current reflection status of *item*.

Description

This system procedure reflects an item vertically, returning the current reflection-and-rotation status of the item.

g2-restore-item-to-normal-size

Restores an item to its default size.

Synopsis

g2-restore-item-to-normal-size
(*item*: class item)

Argument	Description
item	The item whose size you want to restore to its default.

Description

This procedures restores an item whose size has been changed, interactively or programmatically, to its default size, and is the programmatic equivalent of the Restore to Normal Size option available when you choose the change size item menu choice.

g2-set-reflection-and-rotation

Sets an item's current reflection and rotation status.

Synopsis

g2-set-reflection-and-rotation
(*item*: class item, *reflection-and-rotation*: symbol)
-> *reflection-rotation*: symbol

Argument	Description
item	The item whose reflection and rotation status you wish to set.
reflection-and-rotation	The symbol of the reflection and rotation to set.

Return Value	Description
reflection-rotation	A symbol of the current reflection and rotation status of *item*.

Description

This system procedure sets the reflection and rotation status of an item to the status you provide as a symbol, returning the current reflection and rotation status.

Setting an item's reflection and rotation status is relative to its normal position, and is not cumulative. For example, if an item currently has a rotation status of clockwise-90 and you specify that rotation in the g2-set-reflection-and-rotation system-procedure, the system procedure returns clockwise-90 as the current status, but no visual change occurs.

Examples

A procedure that uses the g2-get-reflection-and-rotation and g2-set-reflection-and-rotation system procedures to check whether an item is rotated at clockwise-90, and rotates it if it is not, is:

```
rotate-object-90(obj: class object) = (symbol)
status: symbol;
begin
     status = call g2-get-reflection-and-rotation(obj);
     if status /= the symbol clockwise-90
          then status =
               call g2-set-reflection-and-rotation(obj, the symbol clockwise-90);
     return status
end
```

Notice that when setting a reflection and rotation status (as the example shows), you must precede the reflection and rotation symbol name argument with the symbol statement.

Selection API

The selection API provides a way for KB developers to access and control various aspects of the selection-style user interface. It consists of a set of system procedures and a number of hidden attributes.

The API does not return non-items or non-kb-workspaces. It also obeys the may-refer-to-inactive-items evaluation attribute. For this reason, the selection you see on the screen might be larger than the one returned by the API. For the case of displaying the hidden-attributes table of a G2 window, since there is no context available from which to get evaluation attributes, we assume that it is allowed to show inactive objects.

Selection System Procedures

A workspace can be selected only if it is showing on the given window. An item can be selected only if its workspace can. The window must use the standard user interface style, that is, `-ui standard`.

g2-ui-select
(*selection*: item | sequence, *window*: class g2-window)

Selects the given item or sequence of items in *window*. An error is signalled for any item whose workspace is not showing in the window or which does not have a workspace. *Selection* can also be a workspace, which must be showing.

Here are some examples:

```
g2-ui-select(wire-1, this window)

g2-ui-select(every connection-post upon this workspace, this window)

g2-ui-select(this workspace, this window)

g2-ui-select(sequence(cp-1, wire-1, cp-3), this window)
```

g2-ui-deselect
(*selection*: item | sequence, *window*: class g2-window)

Deselects the given item or sequence of items. *Selection* can be a workspace.

Here are some examples:

```
g2-ui-deselect(cp-1, this window)

g2-ui-deselect(this workspace, this window)

g2-ui-deselect(sequence(cp-1, wire-1, cp-3), this window)
```

g2-ui-deselect-all
(*workspace*: class kb-workspace, *window*: class g2-window)

Deselects all items on the given *workspace*.

For example:

```
g2-ui-deselect-all(this workspace, this window)
```

g2-ui-get-selection
(*workspace*: class kb-workspace, *window*: class g2-window)
-> *selection*: sequence

Returns a sequence of all the selected items on the *workspace* showing in *window*. The first element of the sequence, if any, is guaranteed to be the primary selection.

For example, this API call:

```
g2-ui-get-selection(this workspace, this window)
```

might return a sequence such as this:

```
sequence(cp-1, cp-2, cp-3, connection-post-XXX-1)
```

g2-ui-set-selection
(*selection*: sequence, *window*: class g2-window)
-> *selection*: sequence

Sets the set of selected items on *workspace* to be the elements of the given *selection* sequence and returns the selection. The first element of the sequence, if any, is made the primary selection. All of the items must be on the same workspace, or an error is signaled.

For example:

```
g2-ui-set-selection(sequence(cp-1, cp2), this window)
```

g2-ui-is-selected
(*item*: class item, *window*: class g2-window)
-> truth-value

Returns whether *item* is selected in *window*. Signals an error if the item is not showing. *Item* can be a workspace.

For example:

```
item: class item;
window: class g2-window;
return: truth-value;

return = call g2-ui-is-selected(item, window)
```

g2-ui-get-selected-workspace
(*window*: class g2-window)
-> *workspace*: kb-workspace

Returns the selected workspace, if any, on *window*.

For example:

```
g2-ui-get-selected-workspace(this window)
```

g2-ui-select-workspace
(*workspace*: class kb-workspace, *window*: class g2-window)

Equivalent to g2-ui-select (*workspace*, *window*).

Selection Hidden Attributes

The selection user interface API requires two new hidden read/write attributes of g2-window.

the selected-workspace of *window*
-> no-item | *selected-workspace*: class kb-workspace

Returns the selected kb-workspace on *window* if there is one; otherwise, returns no-item.

Reading this attribute is the same as:

g2-ui-get-selected-workspace(*window*)

Writing this attribute is the same as:

g2-ui-select(*workspace*, *window*)

It can be difficult or impossible to conclude a workspace into this attribute, due to constraints in the G2 compiler and grammar. For that reason, you can conclude a single-element sequence containing the workspace into the attribute. The setter stores the first element of the sequence in that case. For example:

conclude that the selected-workspace of this window = sequence(my-wksp)

is equivalent to:

g2-ui-select(my-wksp, this window)

the selected-items of *window*
-> *selected-items*: sequence

Returns a sequence of the selected items on the selected workspace of *window*.

Reading this attribute is the same as:

g2-ui-get-selection(the selected-workspace of *window*, *window*)

Writing this attribute is the same as:

g2-ui-set-selection(the selected-workspace of *window*, *window*)

Selection Callbacks

This API procedure registers a listener for selection events on a given window, which executes a callback when the event occurs. The procedure returns a handle to the callback, which you use to manage callbacks.

g2-ui-register-selection-callback
(*event*: symbol, *procedure*: class procedure, *user-data*: item-or-value, *window*: class g2-window)
-> *handle*: item-or-value

Event is a symbol naming an event relating to the selection on *window*. The only supported event currently is SELECTION-CHANGED. Whenever the current selection changes on the window, *procedure* is started.

procedure must have the following syntax:

```
my-selection-callback
    (event: symbol, user-data: item-or-value, handle: integer,
    window: class g2-window)
```

where:

event is the symbol SELECTION-CHANGED.

user-data is the same item-or-value supplied above.

handle is the callback handle returned above.

window is the window supplied above.

The current selection can change when an item is selected or deselected on the currently selected workspace, or when the currently selected workspace changes. Note that selecting or deselecting items on a workspace that is *not* selected does not change the current selection.

Multiple changes to the selection are generally coalesced into one call to the callback if they occur within the space of about 100 milliseconds.

For example, this API call registers selection-callback as a procedure to call whenever the selection changes on the given g2-window:

```
g2-ui-register-selection-callback(the symbol SELECTION-CHANGED,
selection-callback, my-wksp, this window)
```

Here is the callback procedure:

```
selection-callback(event: symbol, user-data: item-or-value, handle: integer,
item: class item)
begin
    post "[event] on window [the name of item]";
end
```

Workspace Callbacks

This API procedure registers a listener for workspace events, which executes a callback when the event occurs. The procedure returns a handle to the callback, which you use to manage callbacks.

g2-ui-register-workspace-callback
(*event*: symbol, *procedure*: class procedure, *user-data*: item-or-value, *workspace*: class kb-workspace)
-> *handle*: integer

Registers a listener for workspace events and executes a callback when the event occurs. The procedure returns a handle to the callback, which you use to manage callbacks.

Event is a symbol naming an event relating to *workspace*. The supported events are WORKSPACE-HIDDEN and SCROLLBAR-CHANGE. Whenever the workspace is hidden or the scrollbar of the workspace is changed, *procedure* is started.

Procedure must have the following syntax:

my-workspace-callback
(*event*: symbol, *workspace*: class kb-workspace, *window*: g2-window, *user-data*: item-or-value, *handle*: integer)

where:

event is the symbol WORKSPACE-HIDDEN or SCROLLBAR-CHANGED.

workspace is the kb-workspace supplied above.

user-data is the same item-or-value supplied above.

handle is the callback handle returned above.

window is the window on which the workspace was hidden when the event is WORKSPACE-HIDDEN or on which the scrollbar was changed when the event is SCROLLBAR-CHANGED.

UI Callback Management

These API procedures manage registered UI callbacks. You access the callbacks, using a handle, which you get when you register the callback, using g2-ui-register-selection-callback or g2-ui-register-workspace-callback.

g2-ui-get-callback
(*handle*: integer)
-> *callback*: structure

Returns the registered callback corresponding to *handle*, as a structure.

See g2-ui-get-callbacks for an example.

g2-ui-get-callbacks
(*event*: item-or-value)
-> *callbacks*: sequence

Returns a sequence the handles of all callbacks registered on the given *event*. Specify the symbol all for the event to get all callbacks for all event types.

For example:

```
test-get-callbacks()
result: sequence;
cb: structure;
i: integer;
begin
    result = call g2-ui-get-callbacks(the symbol ALL);
    for i  = each integer in result do
        cb = call g2-ui-get-callback(i);
        post "Callback: [cb]";
    end;
end
```

The resulting callback looks similar to this:

```
structure(HANDLE: 1, ENABLED: true,
EVENT: the symbol SELECTION-CHANGED,
PROCEDURE: SELECTION-CALLBACK, USER-DATA: MY-WKSP,
ITEM: REMOTE-WINDOW-1, OPTIONS: structure())
```

To access the callback handle, refer to the HANDLE attribute of the returned sequence, for example:

```
handle = the HANDLE of cb;
```

g2-ui-disable-callback
(*handle*: item-or-value)

Disables the callback corresponding to *handle*. Disabled callbacks are not called by the system.

Using the g2-ui-get-callbacks example, which accesses a handle from a callback, this example disables the callback corresponding to handle:

```
g2-ui-disable-callback(handle)
```

g2-ui-enable-callback
(*handle*: item-or-value)

Enables the callback corresponding to *handle*.

Using the g2-ui-get-callbacks example, which accesses a handle from a callback, this example disables the callback corresponding to handle:

```
g2-ui-enable-callback(handle)
```

g2-ui-unregister-callback
(*handle*: item-or-value)

Removes and reclaims the callback corresponding to *handle*.

Using the g2-ui-get-callbacks example, which accesses a handle from a callback, this example unregisters the callback corresponding to handle:

```
g2-ui-unregister-callback(handle)
```

Shortcut Bar Views

G2 provides a Windows shortcut bar pane, which you access through Telewindows.

Note Shortcut bars are only supported in Telewindows Next Generation (`twng.exe`).

For examples, see Using Shortcut Bars in Windows Views, Panes, and UI Features in the *G2 Reference Manual*.

g2-ui-create-shortcut-bar
(*folders*: sequence, *callback*: symbol, *options*: structure, *win*: class g2-window)
-> *handle*: integer

Creates a shortcut bar on the given window and returns an integer handle to the shortcut bar.

The *folders* argument is a sequence of structures that describes the folders and their elements. The structure that describes each folder has this syntax:

```
structure
  (label: text,
  items: sequence-of-structures)
```

where:

- label is the text label to display at the top of the shortcut bar.
- items is a sequence of structures that describes each item in the folder.

The structure that describes each item has this syntax:

```
structure
(label: text,
 icon: symbol)
```

where:

- label is the text label to display below each icon. Note that Windows limits the label text to ensure that labels do no overlap. Hovering the mouse over the icon or text displays the full label in a popup if the label is cut off.
- icon is a symbol that names the icon to display in the shortcut bar, which can be a G2 class name, an item instance, or a GMS icon. See Tree Views.

The *callback* is a procedure that gets called when the user interacts with the shortcut bar by selecting an item, right clicking an item, renaming an item, or

closing the shortcut bar. Specify the callback as the symbol none to provide no callback. The syntax of the callback procedure is:

my-shortcut-bar-callback
(*event*: symbol, *win*: class g2-window, *control*: integer, *item*: value, *info*: structure, *user-data*: value)

Argument	Description
event	A symbol naming the event that invoked the callback. The options are: item-selected, right-click, item-renamed, and closed. The callback also responds to the event types described in View Events.
win	The window in which the shortcut bar appears.
control	The integer handle of the shortcut bar.

Argument	Description
item	The text of the label of the selected item in the shortcut bar.
info	For the selected, right-click, item-renamed events, a structure that describes the selected item, as follows: structure (folder: *integer*, item: *integer*) where: • folder is a zero-based index of the folder in which the selected item is defined. • item is a zero-based index of the selected item in the folder. For a description of the *info* structure for the closed event, see the callback procedure in the description of g2-ui-create-html-view. The state attribute in the *info* structure returns docked.
user-data	For the selected, right-click, item-renamed events, a structure that describes the selected item, as follows: structure (label: *text*, icon: *symbol*) where: • label is the text label of the selected item. • icon is the symbol naming the selected item. For the closed event, *user-data* is the symbol none.

The *options* argument is a structure with this syntax:

```
structure
(style: symbol,
 width: integer,
 title: text,
 left: integer,
 top: integer,
 icon: symbol,
 container: symbol-or-integer,
```

neighbor: *value*,
dock: *symbol*,
allow-rename-item: *truth-value*,
allow-drag-item: *truth-value*,
background-color: *symbol*
foreground-color: *symbol*)

where:

- style is one of these symbols: default or listbar. For a description of the different styles, see Using Shortcut Bars in Windows Views, Panes, and UI Features in the *G2 Reference Manual*. Suggestion
- width is the initial width of view window, in pixels.
- title is the text of the shortcut bar title.
- left — The initial position of left side of the shortcut bar, in pixels.
- top — The initial position of top of the shortcut bar, in pixels.
- icon — The icon to use for the docking pane, as a symbol.
- container — One of the following:
  - pane, as a symbol, which displays the view in a docked pane. The default is pane.
  - mdi-child, as a symbol, which displays the view in a floating pane.
  - A dialog handle, as an integer, which displays the view in a dialog. The view is placed in the dialog, replacing the existing control, which must be a label. When container is a dialog handle, the neighbor attribute can be the control ID of another dialog against which to dock, in which case the dock attribute must be the symbol within.
  - The handle of a listbar-style shortcut bar, in which case the neighbor option is the number of the folder within the listbar into which to create the view.
- neighbor — Another pane against which to dock, as a view designation. See g2-ui-is-valid-window-handle for a description of a view designation. The default value is the overall frame.

  To specify that the shortcut bar pane be placed in a tab pane within another pane, specify these options:
  dock: the symbol within, neighbor *h*, where *h* is the pane within which to place the shortcut bar pane.

  To place the shortcut bar within a custom dialog container, specify neighbor as the control-id of a control in the dialog, and specify dock as within.

- dock — The docking edge, which can be one of these symbols: left, top, right, bottom, float, or within. The default value is left.
- allow-rename-item — Allows the item to be renamed. The default is false.
- allow-drag-item — Allows the item to be dragged. The default is false.
- background-color — Specifies the background color of the view.
- foreground-color — Specifies the background color of the view.

g2-ui-manage-shortcut-bar
(*action*: symbol, *handle*: integer, *arg*: value, *win*: class g2-window)

Sends commands to an existing shortcut bar.

Argument	Description
action	The command name to execute. The following commands are supported: • small-icons changes the icon size to use small icons. • large-icons changes the icon size to use large icons. • disable-folder disables all the items in a folder. You specify the folder as an argument. • enable-folder enables all the items in a folder. You specify the folder as an argument. • clear clears all folders and items in a shortcut bar. • destroy deletes the shortcut bar. • background-color changes the background color. • foreground-color changes the foreground color.
handle	The handle to the HTML view as returned by g2-ui-create-shortcut-bar.

Argument	Description
arg	An argument to the command. The only actions that require an argument are: • disable-folder and enable-folder, where *arg* is a zero-based index of the folders as specified in the *folders* argument to g2-ui-create-shortcut-bar. • background-color and foreground-color, where *arg* is a symbol naming a color.
win	The g2-window on which the HTML view was created.

Status Bar Operations

G2 provides a Windows status bar, which you access through Telewindows.

g2-ui-configure-status-bar
(*options*: structure, *win*: class g2-window)

Configures the status bar of the Telewindows window. *Options* is a structure with the following syntax:

structure
(visible: *truth-value*,
callback: *symbol-or-procedure*,
min-height: *integer*,
panes: *sequence*)

where:

- visible — Whether to show or hide the status bar. Set to false to hide the status bar.
- callback — A callback procedure for mouse clicks on the status bar, which has this syntax:

  my-shortcut-bar-callback
  (*event*: symbol, *handle*: integer, *pane-id*: value, *info*: structure, *user-data*: value)

  The options for *event* are left-click, middle-click, right-click, or any combination of these events plus modifiers, for example, control+left-click, shift+right-click, control+alt+left-click, double+left-click.

  The *info* structure contains the mouse X and Y position.

  The *handle* argument is not meaningful for status bars.
- min-height — The minimum height of the status bar, in pixels.
- panes — A sequence of panes in the status bar, where each pane is a structure with the following syntax:

  structure
  (id: *integer* | *symbol* | *text*,
  text: *text*,
  icon: *symbol* | *item*,
  width: *integer* | *symbol*,
  background-color: *symbol* | *truth-value*,
  foreground-color: *symbol* | *truth-value*,
  alignment: *symbol*,
  tooltip: *text*,
  visible: *truth-value*,
  enabled: *truth-value*,

borders: *truth-value*,
user-data: *item-or-value*)

where:

- id — A user-supplied identifier for the pane, which is supplied to other actions. The ID is not required to be unique. The default value is the value of the text attribute. See description below.
- text — A text string to show in the pane. The default is "".
- icon — An icon to show in the pane. The default is false.
- width — The width of the pane in pixels, or the symbol fit, which fits the width to the text. The default is the symbol fit.
- background-color — The background color. The default is false, which means transparent.
- foreground-color — The foreground color. The default is false, which means black.
- alignment — The alignment of the icon and text in the pane. The options are: left, center, or right. The default is the symbol left.
- tooltip — The tooltip to show when hovering the mouse over the pane. The default is "".
- visible — Whether the pane is visible. The default is true.
- enabled — Whether the pane is enabled. The default is true.
- borders — Whether the pane has borders. The default is true.
- user-data — Arbitrary user data attached to the pane, which is returned in the callback. The default is the symbol none.

A pane ID is the name of a pane in the status bar. It is useful to give panes names so they can be referenced without regard to their absolute position in the status bar, which may change as panes are added and removed.

A pane ID can be any integer, a symbol, or a text string. The following integer and symbol values are reserved for use by G2:

- 0 — The built-in documentation-line pane.
- -1 — Indicates beyond the last pane (for ADD-PANE only).
- g2-documentation-line — Another name for the documentation-line pane.
- g2-security-icon — The pane showing the padlock icon, for SSL secured connections.
- g2- — Any other symbol beginning with g2- is reserved for future use.

A pane ID is not required to be unique. In that case, the ID refers to some pane.

g2-ui-manage-status-bar
(*action*: symbol, *arg*: value, *win*: class g2-window)

Manages existing status bars, where *action* is one of these symbolx with corresponding values for *arg*, where applicable:

- hide — Hides the status bar, which is the same as calling g2-ui-hide-status-bar.
- show — Shows the status bar, which is the same as calling g2-ui-show-status-bar.
- set-text — Sets the text of the documentation-line pane in the status bar, which is the same as calling g2-ui-set-status-bar-text. The *arg* is a text value.
- configure — Configures the status bar, which is the same as calling g2-ui-configure-status-bar. The *arg* is a structure, which is the same as the *options* argument to g2-ui-configure-status-bar.
- set-min-height — Sets the minimum height of the status bar, in pixels. The *arg* is an integer.
- set-callback — Sets the callback procedure. The *arg* is a symbol or procedure.
- add-pane — Adds a pane to the status bar. The *arg* is a structure, which is the same as the pane structure for g2-ui-configure-status-bar, which one additional attribute:
  - position: *integer* | *symbol* | *text* — The ID of the pane to insert before. The default is -1, which means add the pane on the end.
- modify-pane — Modifies the pane specified by the ID. The *arg* is a structure, which is the same as the pane structure for g2-ui-configure-status-bar.
- remove-pane — Removes a pane. The *arg* is the ID of the pane to remove or structure(id: *the-id-of-pane-to-remove*).

g2-ui-hide-status-bar
(*window*: class g2-window)

Hides the status bar in a given window.

g2-ui-show-status-bar
(*window*: class g2-window)

Shows the status bar in a given window.

g2-ui-set-status-bar-text
(*string*: text, *window*: class g2-window)

Sets the text of the status bar to the specified string.

System Commands

Use the system-command system procedure to perform a number of commands usually accessible only through the G2 menu system.

g2-system-command

Performs a G2 system command programmatically.

Synopsis

g2-system-command
(*command*: symbol, *win*: class g2-window, *item*: class item,
attribute: symbol)

See the following pages for a table of the acceptable commands and their arguments, and descriptions and examples of how to call this procedure.

Argument	Description
command	Specifies the system command to execute, which are either menu choices or keystroke commands. User-menu choices are not supported. Precede each of these commands with the symbol statement. • General commands: change-mode change-password change-size clone close-telewindows-connection delete describe describe-chaining describe-configuration disable do-not-highlight-invoked-rules dynamic-backward-chaining dynamic-generic-rule-display dynamic-rule-invocation-display edit edit-icon enable enable-all-items help highlight-invoked-rules inspect log-out long-menus network-info new-title-block redo-layout refresh reinstall-authorized-users remove-tracing-and-breakpoints save-kb short-menus show-unsaved-attributes table table-of-hidden-attributes turn-off-all-explanation-caching transfer turn-on-all-explanation-caching

Argument	Description
	• Workspace commands: full-scale normalized-full-scale scale-to-fit maximum-scale-to-fit one-quarter-the-scale four-times-the-scale twenty-percent-smaller twenty-percent-bigger twenty-percent-narrower twenty-percent-wider center-origin shift-left-ten-percent shift-right-ten-percent shift-left-one-percent shift-right-one-percent shift-up-ten-percent shift-down-ten-percent shift-up-one-percent shift-down-one-percent lift-to-top drop-to-bottom circulate-up circulate-down toggle-visible-grid
win	The window from which to invoke the system command. The reason for this argument is that the system command that you invoke is applicable *only* in the window you specify.

Argument	Description
item	Indicates the item to which the system command applies. These system commands require item arguments: describe *item* describe-configuration *item* edit *item* edit-icon *object* enable *item* describe-chaining *variable* disable *item* dynamic-backward-chaining *variable* dynamic-generic-rule-display *variable-or-parameter*, *object* dynamic-rule-invocation-display *rule* redo-layout *g2gl-body* If you are not using one of these system commands, you can enter anything in this argument as long as it meets the type requirement. The system command edit invokes the Text Editor for the attribute that you specify in the next argument. For the edit-icon system command, this argument must evaluate to an object.
attribute	Specifies the attribute to edit. This argument is currently only valid with the edit system command. You can invoke the Text Editor for almost any item attribute, including the text of rules and procedures, but not for compound attributes such as those of charts and trend charts.

Description

This system procedure lets you perform one of a number of G2 commands. The command is executed as if you had selected it interactively from the G2 main menu, a workspace menu, or an item menu, as appropriate.

The system procedure operates within the user mode of the window you specify, so any configurations remain in effect. For example, if the user mode of the current window restricts the user from accessing the Change Mode menu option, this procedure cannot override that configuration.

Before you can use g2-system-command to enable a disabled item, you must authorize the item invoking the procedure to reference inactive and disabled items. To accomplish this, execute the action:

```
conclude that the may-refer-to-inactive-items
of the evaluation-attributes of my-item is true
```

where *my-item* is the item that contains the invocation of g2-system-command. Execute this action at any time before *my-item* first invokes g2-system-command. For information about may-refer-to-inactive-items, see Attribute Access Facility in the *G2 Reference Manual*.

You can also use activatable subworkspaces to programmatically enable and disable items, as described Workspaces in the *G2 Reference Manual*.

Note When using g2-system-command to call the edit command to edit an item, the procedure does not wait until the edit workspace is closed before passing control back to the calling procedure.

When displaying a table for an item, the system command places the table as close to the x-y center of the item as possible, assuming the item is on a workspace that is shown in the given window. If the item is not visible, for example, because it is hidden behind another workspace, the command does not bring the item into view. If the item is off screen in the window, the table appears against the edge of the screen closest to the item. If the item is not on any visible workspace in the window, the table appears in the center of the window.

When cloning and transferring an item, the item must be on a workspace that is shown in the window. If the item is not visible, the command does not bring the item into view. Executing this command transfer the item to the mouse whenever it is over a workspace as if the clone or transfer menu choice had been chosen.

When deleting an item, the command displays the delete confirmation dialog at the center of the window, even if the item is not visible in the window. It is the application's responsibility to make it clear to the user which item is being deleted, as needed.

When changing the size of an item, the item must be on a workspace that is shown in the window. If the item is not visible, the command does not bring the item into view. Executing this command displays the Change Size dialog and places a rectangle around the item as if the change size menu choice had been chosen.

Examples

The following call displays the G2 help menu:

```
call g2-system-command(the symbol help, gen-local-window,
                       gen-local-window, the symbol none)
```

This statement invokes the icon editor for car:

```
call g2-system-command(the symbol edit-icon, gen-local-window, car,
                       the symbol none)
```

This call invokes the editor on the action attribute of an action-button item:

```
call g2-system-command( the symbol edit, gen-local-window,
                       the action-button upon procedure-workspace,
                       the symbol action)
```

This call displays the attribute table for tank-1:

```
tank-1: class item;
win: class g2-window;

call g2-system-command(the symbol table, win, tank-1, the symbol none)
```

Explanation Facilities Example

The following following shows how to display a chaining diagram for a variable that is the value of an attribute.

```
call g2-system-command(the symbol describe-chaining, gen-local-window,
                       the variable giving the gauge-variable-attribute of
                           steel-equipment506)
```

The first argument to g2-system-command is the symbol describe-chaining which is available as a menu choice on the tables of variables on workspaces when the enable-explanation-controls attribute of the Miscellaneous Parameters system table is set to yes. The menu choice does not appear on the subtables of subobjects.

See Explanation Facilities in the *G2 Reference Manual* for information on explanation displays.

Closing a Telewindows Connection

The next example shows how to use this system procedure to close a Telewindows connection. Here, you pass the node name of the Telewindows client to shut down as a text string in an action button. Within the procedure, notice the syntax to compare that name against the value of the g2-window-remote-host-name attribute of a g2-window object ("@"[name]@""). G2 requires this syntax since the attribute value is a string.

```
test-shutdown-telewindows(node: text)
WIN: class g2-window;
TXT: text;
begin
     for WIN = each g2-window
          do
               TXT = the text of the g2-window-remote-host-name of WIN;
               if TXT = "@"[node]@"" then
                    start g2-system-command(
                         the symbol close-telewindows-connection, WIN,
                         WIN, the symbol none)
          end
end
```

Text Operations

Use the text system procedures to access and change the dimensions and fonts of text boxes, and copy text to the clipboard.

g2-copy-text-to-clipboard

g2-get-font-of-text-box

g2-get-maximum-height-of-text-box

g2-get-maximum-width-of-text-box

g2-get-text-extent

g2-set-font-of-text-box

g2-set-maximum-height-of-text-box

g2-set-maximum-width-of-text-box

g2-copy-text-to-clipboard

Copies text to the clipboard.

Synopsis

g2-copy-text-to-clipboard
(*txt*: text, *win*: class g2-window)

Argument	Description
txt	The text to copy.
win	The window from which you wish to copy the text.

Description

This procedure copies *txt* to the clipboard of the host computer of *Win* where it can be retrieved by the host G2, by another G2 process, or by any other application that has editing capability.

g2-get-font-of-text-box

Gets the font size and the x- and y-magnification of any text-box item.

Synopsis

g2-get-font-of-text-box
(*text-box*: class item)
-> *font-size*: symbol, *x-magnification*: quantity, *y-magnification*: quantity

Argument	Description
text-box	The text-box whose font and magnification size you want to obtain. Valid classes for this argument are: statement message free-text borderless-free-text text-inserter Entering an invalid class for this argument causes G2 to signal an error.

Return Value	Description
font-size	A symbol of the text box font size (small, large, or extra-large).
x-magnification	A quantity of the x-magnification of the text box.
y-magnification	A quantity of the y-magnification of the text box.

Description

This system procedure gets the following information about a text-box item of the valid classes listed above:

- The current font-size of the text-box, as a symbol (small, large, or extra-large).
- The current x-magnification of the text-box as a float value.
- The current y-magnification of the text-box as a float value.

g2-get-maximum-height-of-text-box

Returns the maximum allowable height of a text-box.

Synopsis

g2-get-maximum-height-of-text-box
(*text-box*: class free-text)
-> *maximum-height*: integer

Argument	Description
text-box	The text-box whose maximum height you want to obtain. Valid classes for this argument are: statement message free-text borderless-free-text text-inserter

Return Value	Description
maximum-height	An integer indicating the maximum allowable height of the text box in pixels

Description

This procedure returns the maximum allowable height of a text-box.

g2-get-maximum-width-of-text-box

Returns the maximum allowable width of a text-box.

Synopsis

g2-get-maximum-width-of-text-box
text-box: class text-box)
-> *maximum-width*: integer

Argument	Description
text-box	The text-box whose maximum width you want to obtain. Valid classes for this argument are: statement message free-text borderless-free-text text-inserter

Return Value	Description
maximum-width	An integer indicating the maximum width of the text box in pixels.

Description

This procedure returns the maximum allowable width of a text-box.

g2-get-text-extent

Returns width and height information on potential instances of text items based on the text argument and the properties of the class argument.

Synopsis

g2-get-text-extent
(*txt*: text, *text-box-subclass*: symbol)
-> *text-extent-information*: structure

Argument	Description
txt	The text of the item whose extent information you are obtaining.
text-box-subclass	The class of the text item. The applicable classes are: statement message free-text borderless-free-text text-inserter

Return Value	Description
text-extent-information	A structure of integer values that give the text extents.

Description

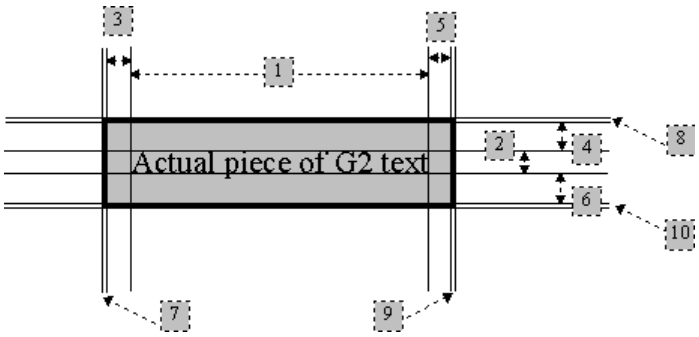
Use this procedure to determine what height and width characteristics an instance of a text class will have when the instance does not override class properties.

This procedures returns a structure that gives the margins, whitespace, and other useful information about text-based classes. The return values for each property of the text item is expressed in pixels, so that you can achieve precise placement of items relative to the actual text.

The returned structure includes ten attributes, each of type integer:

Attribute name and position in structure	Description
width (1)	The width in x of the text bounding box.
height (2)	The height in y of the text bounding box.
left-margin-width (3)	The left margin width.
top-margin-width (4)	The top margin width.
right-margin-width (5)	The right margin width.
bottom-margin-width (6)	The bottom margin width.
left-border-width (7)	The left border width.
top-border-width (8)	The top border.
right-border-width (9)	The right border width.
bottom-border-width (10)	The bottom border width.

This diagram illustrates the returned attributes. The numbers indicate the attribute positions in the returned structure.



g2-set-font-of-text-box

Sets the font size and the x- and y-magnification of any text-box item.

Synopsis

g2-set-font-of-text-box
(*text-box*: class item, *font-size*: symbol,
x-magnification: quantity, *y-magnification*: quantity)
-> *font-size*: symbol, *x-magnification*: quantity, *y-magnification*: quantity

Argument	Description
text-box	The text-box whose font and magnification size you want to set. Valid classes for this argument are: statement message free-text borderless-free-text text-inserter Entering an invalid class for this argument causes G2 to signal an error.
font-size	The font size you want the text-box item from the previous argument to use. Valid symbol font sizes are small, large, and extra-large.
x-magnification	The x (horizontal) magnification of the text-box in the first argument. Enter the magnification as a quantity value (either float or integer). Valid magnification ranges are 0.125 – 4.0. Entering a value of 1 or 1.0 is equivalent to no magnification.
y-magnification	The y (vertical) magnification of the text-box in the first argument. Enter the magnification as a quantity value (either float or integer). Valid magnification ranges are 0.125 – 4.0. Entering a value of 1 or 1.0 is equivalent to no magnification.

Return Value	Description
font-size	A symbol of the font size of the text box (small, large, or extra-large).
x-magnification	A quantity of the x-magnification of the text box.
y-magnification	A quantity of the y-magnification of the text box.

Description

This system procedure lets you set the font size and magnification of a text-box item, which can be one of the classes listed under the description of the first argument. The procedure returns this information:

- The current font-size of the text-box, as a symbol (small, large, or extra-large).
- The current x-magnification of the text-box as a float value.
- The current y-magnification of the text-box as a float value.

Example

Here is a procedure that uses the g2-get-font-of-text-box and g2-set-font-of-text-box procedures to get the font size and magnification of a instance of borderless-free-text and then set the font to large and double the x and y magnifications.

```
get-and-set-font-size-and-magnification(free-text: class borderless-free-text)
     = (symbol, quantity, quantity)
x-magnification, y-magnification: quantity;
font-size = symbol;
begin
     font-size, x-magnification, y-magnification =
          call g2-get-font-of-text-box(free-text);
     font-size, x-magnfication, y-magnification =
          call g2-set-font-of-text-box(free-text, the symbol large,
               (x-magnification * 2), (y-magnification * 2));
     return font-size, x-magnification, y-magnification
end
```

g2-set-maximum-height-of-text-box

Sets the maximum allowable height of a text-box.

Synopsis

g2-set-maximum-height-of-text-box
(*text-box*: class item, *max-height:* integer)

Argument	Description
text-box	The text-box whose maximum height you want to specify. Valid classes for this argument are: statement message free-text borderless-free-text text-inserter
max-height	An integer from 0 to 6500 indicating the maximum allowable height in pixels.

Description

This procedure sets the maximum height of a text box to a new value.

g2-set-maximum-width-of-text-box

Sets the maximum allowable width of a text-box.

Synopsis

g2-set-maximum-width-of-text-box
(*text-box*: class item, *max-width:* integer)

Argument	Description
text-box	The text-box whose maximum width you want to specify. Valid classes for this argument are: statement message free-text borderless-free-text text-inserter
max-width	An integer from 0 to 6500 indicating the maximum allowable width in pixels.

Tree Views

G2 provides a Windows tree view pane, which you access through Telewindows.

Note Tree views are only supported in Telewindows Next Generation (`twng.exe`).

For examples, see Using Tree Views in Windows Views, Panes, and UI Features in the *G2 Reference Manual*.

Note The following system procedures also work for property grids: g2-ui-select-node, g2-ui-expand-node, g2-ui-collapse-node, g2-ui-modify-node, g2-ui-insert-node, g2-ui-delete-node.

g2-ui-create-tree-view
(*title*: text, *callback*: symbol, *options*: structure, *win*: class g2-window)
-> *handle*: integer

Creates a tree view in *win* with *title* as the title text, which appears both as a vertical tab and a horizontal title bar. This procedure returns an integer handle to the tree view.

You can register a *callback*, which is the name of a procedure to call when various events occur. The syntax for the callback procedure is:

my-tree-view-callback
(*event*: symbol, *win*: class g2-window, *control*: integer, *item*: value, *info*: structure, *user-data*: value)

Argument	Description
event	A symbol naming the event that invoked the callback. The options are: • select — A tree node was selected. • left-click — The user clicked the left mouse button on a tree node. • right-click — The user clicked the right mouse button on a tree node. • double+left-click — The user double-clicked the left mouse button on a tree node. • Modifier keys in combination with clicking the left mouse button, for example, shift+left-click • closed — The tree view was closed. • key-pressed — A key was pressed. The key name is in the key attribute of *info*. Only keys not processed by the tree view are sent. • node-expanded — A tree node was expanded. • node-collapsed — A tree node was collapsed. The callback also responds to the event types described in View Events.
win	The window in which the tree view appears.
control	The integer handle of the tree view.
item	The text of the label of the selected item in the tree view.

Argument	Description
info	For the select and right-click events, a structure that describes the X-Y location of the mouse click on the selected item, as follows: structure (x: *integer*, y: *integer*) For a description of the *info* structure for the closed event, see the description of the callback for g2-ui-create-html-view. The state attribute in the *info* structure returns docked.
user-data	For the select and right-click events, a structure that describes the tree for the selected item. For the closed event, *user-data* is the symbol none.

The *options* argument is a structure with this syntax:

```
structure
(left: integer,
 top: integer,
 icon: symbol,
 height: integer,
 width: integer,
 default-icon: symbol,
 default-leaf-icon: symbol,
 default-folder-icon: symbol,
 tree: item-or-value,
 container: symbol-or-integer,
 neighbor: value,
 dock: symbol,
 right-click-selects-note: truth-value)
```

where:

- left — The initial position of left side of the tree view window, in pixels.
- top — The initial position of top of the tree view window, in pixels.
- icon — A symbol naming an icon to use for the docking pane. You can specify the class name of the icon to display, as a symbol, an item instance, or one of the built-in GMS icons. When specifying an item instance, the displayed icon includes any color and orientation changes, and any updates if the item instance changes.
- height — The initial height in pixels, using modulo docking. The default is 240.

- width — The initial width in pixels, modulo docking. The default is 240.
- default-icon — The default icon for all nodes in the tree view. See icon for possible values.

  For a list of GMS icons, see the description of push-button in Custom Windows Dialogs in the *G2 Reference Manual*.
- default-leaf-icon — The default icon for nodes without children.
- default-folder-icon — The default icon for nodes with children.
- tree — The initial contents of the tree view. See the description of *tree-view-node* in g2-ui-populate-tree-view.
- container is one of:
  - pane, as a symbol, which displays the view in a docked pane. The default is pane.
  - mdi-child, as a symbol, which displays the view in a floating pane.
  - A dialog handle, as an integer, which displays the view in a dialog. The view is placed in the dialog, replacing the existing control, which must be a label. When container is a dialog handle, the neighbor attribute can be the control ID of another dialog against which to dock, in which case the dock attribute must be the symbol within.
  - The handle of a listbar-style shortcut bar, in which case the neighbor option is the number of the folder within the listbar into which to create the view.
- neighbor — Another pane against which to dock, as a view designation. See g2-ui-is-valid-window-handle for a description of a view designation. The default value is the overall frame.

  To specify that the view pane be placed in a tab pane within another pane, specify these options: dock: the symbol within, neighbor *h*, where *h* is the pane within which to place the view pane. When container is a dialog handle, the neighbor attribute can be the control ID of another dialog against which to dock.
- dock — The docking edge, which can be one of these symbols: left, top, right, bottom, float, or within. The default value is left.
- right-click-selects-node — Whether right-clicking a node in the tree view automatically selects the node. The default is true. Set to false to disable automatic selection when right-clicking the node.

g2-ui-clear-tree-view
(*handle*: integer, *win*: class g2-window)

Clears all items from the tree view specified by *handle* in *win*. The tree view still exists.

g2-ui-destroy-tree-view
(*handle*: integer, *win*: class g2-window)

Permanently removes the tree view specified by *handle* from *win*.

g2-ui-hide-tree-view
(*handle*: integer, *win*: class g2-window)

Hides the tree view specified by *handle* in *win*, as if the user had clicked the close button.

g2-ui-populate-tree-view
(*handle*: integer, *tree*: structure, *win*: class g2-window)

Populates the tree view specified by *handle* with a set of items given by *tree*, which is a structure with this format:

```
structure
    (item-or-name: tree-view-node,
    children: sequence-of-structures)
```

where:

- item-or-name is the top-level node in the tree view.
- children is the children of the top-level node in the tree view.

In the *tree* structure, *tree-view-node* is one of the following:

- A text, which is considered the label, tooltip, and item-or-name.
- A symbol, whose string is taken as the label, tooltip, and item-or-name.
- An item, where the name of the item becomes the label, and the item itself is the item-or-name.
- A structure with this format:

```
(structure
    (item-or-name: item-or-text,
    children: sequence-of-structures,
    icon: symbol,
    selected-icon: symbol,
    label: text,
    tooltip: text,
    expand: truth-value,
    bold: truth-value)
```

where:

- item-or-name — An item, string, or symbol. No default.
- children — A sequence of tree view nodes. The default is sequence().
- icon — The icon to show when node is not selected. You can specify the class name of the icon to display, as a symbol, an item instance, or

one of the built-in GMS icons. When specifying an item instance, the displayed icon includes any color and orientation changes, and any updates if the item instance changes.

- selected-icon — The icon to show when the node is selected, which can be a class name, an item instance, or a GMS icon. The selected-icon defaults to the value of icon.
- label — The label to display. The default is the item-or-name.
- tooltip — The tooltip for the node. The default is the label.
- expand — A truth-value that determines whether to initially expand the node. The default is false.
- bold — A truth-value that determines whether to display the node's label in bold face. The default is false.

In the *tree* structure, *sequence-of-structures* is a sequence of structures with this format:

```
sequence
    (structure
        (item-or-name: item-or-text,
        children: sequence-of-structures,
        icon: symbol,
        selected-icon: symbol,
        label: text,
        tooltip: text,
        expand: truth-value,
        bold: truth-value)
    . . . )
```

The *tree* structure can also include any additional user-defined attributes, for example an id to uniquely identify the item in the tree. The callback returns the user-defined attributes when the tree node is selected.

The *tree* structure can be nested as many levels deep as needed to describe the tree.

If a node in a tree view is an item, not a structure, then the item is transformed into structure (item-or-name: *item*) before being passed as the *user-data* argument to the tree view's callback procedure. This is necessary because the callback procedure for all views must declare their *user-data* argument to be of type value.

Icons displayed in the tree view are always scaled to a size of 16x16 pixels. Some icons, particularly those with a very high aspect ratio, may be rendered unrecognizable by this scaling. For example, a uil-goto-workspace-button instance with a label is actually a very wide and narrow icon, since the label is part of the icon. When compressed to 16x16, the icon looks strange and the text is illegible. In this case, you may prefer to use the generic class icon, which

has an empty label. To use the generic class icon for a tree view node, specify the symbol naming the class as the node's icon option, for example, icon: the symbol uil-goto-workspace-button.

To create a tree view that is the class hierarchy of a particular class, you can use the g2-get-class-hierarchy system procedure by passing in the class to use as the root of the tree view. You can also construct your own tree structure.

g2-ui-select-tree-view-item
(*handle*: integer, *item*: text, *win*: class g2-window)

Selects an element from the tree view specified by *handle* in *win*, expanding the tree view as necessary to show the *item*.

g2-ui-show-tree-view
(*handle*: integer, *win*: class g2-window)

Shows the tree view specified by *handle* in *win*. The tree view is shown at the same level of expansion that it had when it was hidden.

g2-ui-manage-tree-view
(*action*: symbol, *handle*: integer, *arg*: value, *win*: g2-window)

Performs an action on the tree view specified by *handle*. The options for *action* are:

- hide — Hides the tree view.
- select — Selects the tree view pane.
- selected-item — Selects an item in the tree view, where *arg* is the same as the *item* argument to g2-ui-select-tree-view-item.
- populate — Populates a tree view, where *arg* is the same as the *tree* argument to g2-ui-populate-tree-view.
- clear — Clears all nodes in the tree view.
- destroy — Deletes the tree view.

g2-ui-select-node
(*handle*: integer, *node-designation*: item-or-value, *win*: class g2-window)

Selects the designated node, if found, in the tree view specified by *handle*. The *node-designation* argument is an item-or-value used to specify an existing node in a tree view. A *node-designation* is interpreted based on its type, as follows:

- truth-value — The root of the tree (either true or false).
- integer — The unique handle of the node, as returned by g2-ui-insert-node.
- text — The label of the node, which must match exactly, including case.

– item — The item-or-name of the node, as specified by g2-ui-populate-tree-view.

g2-ui-collapse-node
(*handle*: integer, *node-designation*: item-or-value, *win*: class g2-window)

Collapses the designated node, if found and if it has children, in the tree view specified by *handle*. See g2-ui-select-note for a description of *node-designation*.

g2-ui-expand-node
(*handle*: integer, *node-designation*: item-or-value, *win*: class g2-window)

Expands the designated node, if found and it has children, in the tree view specified by *handle*. See g2-ui-select-note for a description of *node-designation*.

g2-ui-delete-node
(*handle*: integer, *node-designation*: item-or-value, *win*: class g2-window)

Deletes the designated node, if found, in the tree view specified by *handle*. See g2-ui-select-note for a description of *node-designation*.

g2-ui-insert-node
(*handle*: integer, *node-to-insert*: item-or-value,
reference-node-designation: item-or-value, *options*: structure,
win: class g2-window)
-> *handle*: integer

Inserts a new node into the tree view specified by *handle*, as a child or sibling of the *reference-node-designation*. See g2-ui-select-node for a description of *node-designation*.

Note that the inserted node may include children, so an entire subtree can be inserted in one call. However, only the handle of the top-most node is returned.

The *options* structure has syntax:

structure
(position: *symbol*)

where *symbol* is one of these symbols:

- first — Insert as the first child of the reference node.
- last — Insert as the last child of the reference node. This is default.
- after — Insert just after the reference node, as a sibling.

g2-ui-modify-node
(*handle*: value, *node-designation*: item-or-value, *changes*: structure,
window: class g2-window)

Modifies the designated node, if found, in the tree view specified by *handle*. The *node-designation* argument is an item-or-value used to specify an existing node in a tree view.

The *changes* structure has the following syntax:

structure
(label: *text*,
tooltip: *text*
icon: *item-or-value*
selected-icon: *item-or-value*
bold: *truth-value*
expand: *truth-value*)

Note that changes are *not* reflected in the node structure received by the tree view's callback (if any), because that structure is read-only. The item argument, however, does reflect the current label.

Trend-Chart Operations

Use the trend-chart system procedures to perform trend-chart component operations.

g2-add-trend-chart-component

g2-delete-trend-chart-component

g2-get-text-of-trend-chart-component

g2-set-text-of-trend-chart-component

g2-add-trend-chart-component

Adds a new **component** to any trend chart **compound attribute** that supports multiple components.

Synopsis

g2-add-trend-chart-component
(*trend-chart*: class trend-chart, *attribute-name*: symbol)

Argument	Description
trend-chart	The trend chart to which you wish to add a new component.
attribute-name	The name of the trend-chart attribute, which can be one of the following symbols: plots value-axes point-formats connector-formats

Description

This system procedure adds a new component subtable to the compound attribute you specify in the second argument.

Examples

An example of using the g2-add-trend-chart-component to add a plot component to a trend chart and then using g2-set-text-of-trend-chart-component to set the use-local-history? attribute to no, and to change the expression attribute to evaluate to a variable, is:

```
add-change-new-component(tc: class trend-chart)
begin
    { Add a new plot to the trend chart. }
    call g2-add-trend-chart-component(tc, the symbol plots);

    { Change the local history to "no" to plot from a variable. }
    call g2-set-text-of-trend-chart-componen(tc, the symbol plots,
        the symbol Use-local-history@?, 2 "no")

    { Enter the name of the variable, Q2, in the expression attribute of the
        new plot. }
    call g2-set-text-of-trend-chart-component(tc, the symbol plots,
        the symbol expression, 2, "Q2")
end
```

g2-delete-trend-chart-component

Deletes a trend chart component.

Synopsis

g2-delete-trend-chart-component
(*trend-chart*: class trend-chart, *attribute-name*: symbol,
component-indicator: item-or-value)

Argument	Description
trend-chart	The trend chart from which you wish to delete a component.
attribute-name	The name of the trend-chart component, which can be one of the following symbols: plots value-axes point-formats connector-formats
component-indicator	The specific component to delete. You can enter either the name of the component (if you have provided one) as a symbol, or the component reference number as an integer.

Description

This system procedure deletes the trend chart component that you specify in the third argument.

You cannot delete the last component of any compound attribute that permits multiple components, or of a compound attribute that consists of only a single component.

Example

A statement that uses the g2-delete-trend-chart-component procedure to delete plot 2, is:

```
call g2-delete-trend-chart-component(tc, the symbol plots, 2)
```

g2-get-text-of-trend-chart-component

Gets the text of a trend chart's component attribute, returning the information as a text value.

Synopsis

g2-get-text-of-trend-chart-component
(*trend-chart*: class trend-chart, *attribute-name*: symbol,
component-attribute-name: symbol,
component-indicator: item-or-value)
-> *component-text*: text, *component-default*: truth-value

Argument	Description
trend-chart	The trend chart whose component attribute information you wish to obtain.
attribute-name	The name of the trend chart compound attribute, which can be one of the following symbols: plots value-axes time-axis point-formats connector-formats trend-chart-format
component-attribute-name	The component attribute of the attribute you indicate in the previous argument. For instance, if you enter plots as the attribute name, you can enter the name of any of the plot component attributes for this argument. Providing a component attribute name that is not an attribute of the component signals an error such as: Unknown trend-chart component specified.
component-indicator	The component reference. You can enter either the component name (if you have provided one) as a symbol, or the component reference number as an integer. For compound attributes that consist of a single component, time-axis and trend-chart-format, enter false for this argument.

Return Value	Description
component-text	The text of a component attribute.
component-default	A truth value which is true if *component-text* specifies the default value for the component attribute or false if it does not.

Description

This system procedure accesses a trend chart component attribute programmatically. It returns the text of a component attribute, and a value of true if the text is the default value for that particular component attribute.

Since trend charts are comprised of compound attributes (those made up of one or more components), their attribute values consist of complex **annotations**. To change even a single character of a trend chart component requires that you first get the complete textual annotation of the attribute so that you can replace it with the new value.

This procedure eliminates the need of first getting the entire text of a compound attribute, such as the text of the plots of my-trend-chart, and then decomposing that text to find the annotation describing a particular component attribute that you want to review or change.

Use this system procedure to get the text of a component attribute, and the g2-set-text-of-trend-chart-component to change it.

Examples

See g2-set-text-of-trend-chart-component.

g2-set-text-of-trend-chart-component

Changes the text of a trend chart's component attribute.

Synopsis

g2-set-text-of-trend-chart-component
(*trend-chart*: class trend-chart, *attribute-name*: symbol,
component-attribute-name: symbol,
component-indicator: item-or-value, *txt*: text)

Argument	Description
trend-chart	The trend chart whose component attribute information you wish to change.
attribute-name	The name of the trend chart compound attribute, which can be one of the following symbols: plots value-axes time-axis point-formats connector-formats trend-chart-format
component-attribute-name	The component attribute of the attribute you indicate in the previous argument. For instance, if you enter plots as the attribute name, you can enter the name of any of the plot component attributes for this argument. Providing a component attribute name that is not an attribute of the component signals an error such as: Unknown trend-chart component specified.

Argument	Description
component-indicator	The component reference. You can enter either the component name (if you have provided one) as a symbol, or the component reference number as an integer. For compound attributes that consist of a single component, time-axis and trend-chart-format, enter false for this argument.
txt	The new value of the component attribute you are changing. To revert to the default value for the component attribute you are changing, enter this argument as: "link to default"

Description

This system procedure changes a single trend chart component attribute value programmatically.

Trend charts have compound attributes (those made up of one or more components), so their attribute values consist of complex annotations. To change even a single character of a trend chart component requires that you first get the complete textual annotation of the attribute so that you can replace it with the new value.

This system procedure eliminates the need to replace the entire text of a compound attribute.

Examples

A procedure that uses the g2-get-text-of-trend-chart-component to obtain the current value of the range-mode component, then calls g2-set-text-of-trend-chart-component to change the range-mode component attribute to fixed, is:

```
change-trend-chart(tc: class trend-chart) = (text, text)
previous-component-text, new-component-text: text;
begin
    previous-component-text =
        call g2-get-text-of-trend-chart-component(tc, the symbol value-axes,
            the symbol range-mode, the symbol cos-value);
    call g2-set-text-of-trend-chart-component(tc, the symbol value-axes,
        the symbol range-mode, the symbol cos-value, "fixed");
    new-component-text =
        call g2-get-text-of-trend-chart-component(tc, the symbol value-axes,
            the symbol range-mode, the symbol cos-value);
    return previous-component-text, new-component-text
end
```

Window and Workspace Operations

Use the window and workspace system procedures to access and change workspace and window properties.

g2-drop-workspace-behind

Drops one workspace behind another.

Synopsis

g2-drop-workspace-behind
(*ws-1*: class kb-workspace, *ws-2*: class kb-workspace,
win: class g2-window)

Argument	Description
ws-1	The workspace currently on top of another, that is to be placed behind the other workspace.
ws-2	The workspace currently beneath the other. This workspace will end up on top of workspace-1.
win	The window in which the workspaces are visible.

Description

Causes G2 to position one workspace behind another. The system procedure accepts two workspaces as its arguments, reversing their stacking order. This procedure does not return a value; it returns as soon as it drops one workspace behind another.

The procedure works as follows. If the layer number of *ws-1* is higher than *ws-2* (it is beneath if the workspaces overlap), nothing happens.

If the layer number of *ws-1* is less than *ws-2*, then G2 changes the layer position of *ws-1* to that of *ws-2* (the layer number of *ws-1* is then higher than *ws-2*). If the two workspaces overlap, *ws-1* then appears below *ws-2*. G2 increments the layer number of *ws-1* by 1, as it does the layer numbers of any other workspaces with layer numbers between *ws-1* and *ws-2*.

g2-drop-workspace-to-bottom

Drops a workspace to the bottom of the current stacking order.

Synopsis

g2-drop-workspace-to-bottom
(*wksp*: class kb-workspace, *win*: class g2-window)

Argument	Description
wksp	The workspace you wish to place at the bottom of the current stacking order.
win	A given window in which the workspace to drop is displaying.

Description

This system procedure causes G2 to position the workspace passed to the system procedure at the bottom of the workspace stacking order, providing a programmatic version of the interactive drop to bottom workspace menu choice. This procedure does not return a value; it returns as soon as the workspace is dropped to the bottom.

g2-get-workspace-layer-position

Returns the layer position of a workspace.

Synopsis

g2-get-workspace-layer-position
(*wksp*: class kb-workspace, *win*: class g2-window)
-> *workspace-layer*: integer

Argument	Description
wksp	The workspace whose layer position you wish to obtain.
win	A given window in which the workspace is displaying.

Return Value	Description
workspace-layer	An integer indicating the layer position of the given workspace.

Description

Each workspace on a given G2 window that is not hidden has a designated layer position, indicating its layer in a stack of workspaces. As workspaces are moved, added to or deleted from the window, the layer numbers are adjusted.

This system procedure gets the current layer number of any visible workspace, returning an integer index indicating the current stacking order within the window. The workspace with the lowest layer number is at the top of the stack. For instance, if three workspaces have the layer numbers 5, 3, and 0, the one whose layer is 0 is the topmost workspace.

This system procedure sets the physical layer of the workspace in the stack, whereas the layer-position hidden attribute determines the number of objects on a workspace, where a new, unnamed workspace has a layer-position of 0. The layer-position gets incremented each time the workspace changes, for example, naming a workspace sets the layer-position to 1, placing a procedure on the workspace sets the layer-position to 2, and so on.

g2-item-is-showing-in-window

Returns true if an item is showing in a given g2-window.

Synopsis

g2-item-is-showing-in-window
(*item*: item-or-value, *client*: class ui-client-item)
-> *showing-in-window*: truth-value

Argument	Description
item	The item that you wish to determine is showing in an given window.
client	A g2-window or ui-client-session item representing the specific window in which to determine whether the item is showing.

Return Value	Description
showing-in-window	A truth-value that is true if the item is showing in the window specified by *client*, or false if it is not.

Description

This system procedure determines whether an item is showing in a given window, returning true if the item is showing, or false if it is not.

The term showing does not necessarily indicate that the item is visible within a window. The item that you pass to the system procedure could be hidden beneath another item, or a workspace obscured by other workspaces. The system procedure returns true whenever the workspace, either the one you pass to the system procedure directly, or the one on which the item you pass to the procedure resides, is not hidden in the given G2 window.

Example

A procedure that checks to see if a workspace is visible in the window and, if so, gets the layer number of the workspace, and displays it for the operator, is:

```
get-workspace-layers(window: class g2-window)
WS: class kb-workspace;
visible: truth-value;
layer: integer;
begin
    for WS = each kb-workspace
        do
            visible = call g2-item-is-showing-in window(WS, window);
            if visible then
                begin
                    layer = call g2-get-workspace-layer-position(WS,
                                window);
                    post "[the name of WS] has layer number [layer]"
                end
        end
end
```

g2-lift-workspace-in-front-of

Lifts one workspace in front of another.

Synopsis

g2-lift-workspace-in-front-of
(*ws-1*: class kb-workspace, *ws-2*: class kb-workspace,
win: class g2-window)

Argument	Description
ws-1	The workspace currently underneath *ws-2* which is to be placed in front of *ws-2*.
ws-2	The workspace currently in front of *ws-1* another. This workspace will end up beneath *ws-1*.
win	The window in which the workspaces are visible.

Description

This system procedure causes G2 to position one workspace in front of another. It accepts two workspaces as its arguments, reversing their stacking order. This procedure does not return a value; it returns as soon as G2 lifts one workspace in front of the other.

The procedure works as follows. If the layer number of *ws-1* is less than *ws-2* (it is on top if the workspaces overlap), nothing happens.

If the layer number of *ws-1* is higher than *ws-2*, then G2 changes the layer position of *ws-1* to that of *ws-2*. if the two workspaces overlap, *ws-1* then appears on top of *ws-2*. g2 increments the layer number of *ws-2* by 1, as it does the layer numbers of any other workspaces with layer numbers between *ws-1* and *ws-2*.

g2-lift-workspace-to-top

Lifts a workspace to the top of the current stacking order.

Synopsis

g2-lift-workspace-to-top
(*wksp*: class kb-workspace, *in-window*: class g2-window)

Argument	Description
wksp	The workspace you wish to place at the top of the current stacking order.
win	A given window in which the workspace to lift is displaying.

Description

This system procedure causes G2 to position the workspace that you pass to it at the top of the workspace stacking order. It is effective only for visible workspaces, providing a programmatic equivalent of the interactive lift to top workspace menu choice. This procedure does not return a value; it returns as soon as G2 lifts the workspace to the top.

g2-set-workspace-layer-position

Sets the layer position of a workspace.

Synopsis

g2-set-workspace-layer-position
(*wksp*: class kb-workspace, *win*: class g2-window, *position*: integer)
-> *stacking-order*: integer

Argument	Description
wksp	The workspace whose current layer position you wish to change to a specific position.
win	A given window in which the workspace is displaying.
position	The new layer position for the workspace.

Return Value	Description
stacking-order	An integer of the current stacking order of the workspace within the g2-window.

Description

This system procedure sets the current layer number of a workspace, returning an integer index indicating the current stacking order within the window. The workspace with the lowest layer number is at the top of the stack. For instance, if three workspaces have the layer numbers 5, 3, and 0, the one whose layer is 0 is the topmost workspace.

Everything that is visible in the given window has a relative layer number. G2 sets the workspace layer position as close as possible to the position you request. If you request a layer position already in use by another workspace, the system procedure renumbers the layer positions of all workspaces in conflict with the request.

This system procedure sets the physical layer of the workspace in the stack, whereas the layer-position hidden attribute determines the number of objects on a workspace, where a new, unnamed workspace has a layer-position of 0. The layer-position gets incremented each time the workspace changes, for example, naming a workspace sets the layer-position to 1, placing a procedure on the workspace sets the layer-position to 2, and so on.

g2-x-in-window

Converts the workspace units of an item's x-position to window units.

Synopsis

g2-x-in-window
(*x-in-workspace*: integer: *wksp*: class kb-workspace, *win*: class g2-window)
-> *x-position-in-window*: integer

Argument	Description
x-in-workspace	The item-x position of any item.
wksp	The workspace on which the item resides.
win	The window of the workspace.

Return Value	Description
x-position-in-window	An integer of the workspace x-position corresponding to *x-in-workspace*.

Description

This system procedure returns an integer value that you can use as the x-position for showing a workspace in the G2 window. Use this system procedure in conjunction with the corresponding g2-y-in-window, to position a workspace in relation to an item in a G2 window.

Note Use this procedure in conjunction with g2-y-in-window.

g2-y-in-window

Converts the workspace units of an item's y-position to window units.

Synopsis

g2-y-in-window
(*y-in-workspace*: integer: *wksp*: class kb-workspace, *win*: class g2-window)
-> *y-position-in-window*: integer

Argument	Description
y-in-workspace	The item-y position of any item.
wksp	The workspace on which the item resides.
win	The window of the workspace.

Return Value	Description
y-position-in-window	An integer of the workspace y-position corresponding to *y-in-workspace*.

Description

This system procedure returns an integer value that you can use as the y-position for showing a workspace in the G2 window. Use this system procedure in conjunction with the corresponding g2-x-in-window, to position a workspace in relation to an item in a G2 window.

Note Use this procedure in conjunction with g2-x-in-window.

g2-x-in-workspace

Translates an x coordinate in a given window into an item-x coordinate on the given workspace.

Synopsis

g2-x-in-workspace
(*x-in-window*: integer, *wksp*: class kb-workspace, *win*: class g2-window)
-> *item-x-in-workspace*: integer

Argument	Description
x-in-window	The x-coordinate in the given window to translate.
wksp	The workspace on which the x-coordinate will correspond.
win	The window in which the original x-coordinate exists.

Return Value	Description
item-x-in-workspace	An integer of the coordinate specified by *x-in-window*, in window units.

Description

This system procedure translates any x-coordinate in a window to corresponding workspace units. Use this system procedure when you know a position in a window at which you wish to position an item upon a workspace.

Note This system procedure is the inverse of the procedure g2-x-in-window.

g2-y-in-workspace

Translates a y coordinate on a given window into an item-y coordinate on the given workspace.

Synopsis

g2-y-in-workspace
(*y-in-window*: integer, *wksp*: class kb-workspace, *win*: class g2-window)
-> *item-y-in-workspace*: integer

Argument	Description
y-in-window	The y-coordinate in the given window to translate.
wksp	The workspace on which the y-coordinate will correspond.
win	The window in which the original y-coordinate exists.

Return Value	Description
item-y-in-workspace	An integer of the coordinate specified by *y-in-window*, in window units.

Description

This system procedure translates any y-coordinate in a window to corresponding workspace units. Use this system procedure when you know a position in a window at which you wish to position an item upon a workspace.

Note This system procedure is the inverse of the procedure g2-y-in-window.

Example

A procedure that gets and converts the item-x and item-y coordinates of an item, and then shows the item's subworkspace in relation to that position in the G2 window:

```
test-x-y-in-window(station: class station, ws: class kb-workspace,
    window: class g2-window)
x-in-window, y-in-window: integer;
begin
    x-in-window =
        call g2-x-in-window(the item-x-position of station, ws, window);
    y-in-window =
        call g2-y-in-window(the item-y-position of station, ws, window);
    show the subworkspace of station with its top left corner at
        (x-in-window + 10, y-in-window - 20) in the screen
end
```

g2-x-scale-of-workspace-in-window

Returns the x-scale of a workspace.

Synopsis

g2-x-scale-of-workspace-in-window
(*wksp*: class kb-workspace, *win*: class g2-window)
-> *current-x-scale*: float

Argument	Description
wksp	The workspace whose x-scale you wish to obtain from the system procedure.
win	The window on which you wish to check the x-scale of the workspace.

Return Value	Description
current-x-scale	A float of the current x-scale of the workspace.

Description

This system procedure returns the x-scale of the workspace in the window as a float value. Use this procedure in conjunction with the g2-y-scale-of-workspace-in-window, described next, to obtain the x-scale so that you can reset it to the desired scale.

Examples

An example of this system procedure is included in the description for g2-y-scale-of-workspace-in-window.

g2-y-scale-of-workspace-in-window

Returns the y-scale of a workspace.

Synopsis

g2-y-scale-of-workspace-in-window
(*wksp*: class kb-workspace, *win*: class g2-window)
-> *current-y-scale*: float

Argument	Description
wksp	The workspace whose y-scale you wish to obtain from the system procedure.
win	The window on which you wish to check the y-scale of the workspace.

Return Value	Description
current-y-scale	A float of the current y-scale of the workspace.

Description

This system procedure returns the y-scale of the workspace in the window as a float value. Use this procedure in conjunction with the g2-x-scale-of-workspace-in-window, described previously, to determine the x-scale so that you can reset it to the desired scale.

g2-ui-get-dialog-base-units

Returns two float values giving the horizontal & vertical dialog base units in pixels.

Synopsis

g2-ui-get-dialog-base-units
(*win*: class g2-window)
-> *hbu*: float, *vbu*: float

Argument	Description
win	The window on which you wish to get the dialog base units.

Return Value	Description
hbu	A float giving the horizontal dialog base units in pixels.
vbu	A float giving the vertical dialog base units in pixels.

Description

This system procedure returns the horizontal & vertical dialog base units (DLUs in pixels). The horizontal & vertical dialog base units can be used when dynamically sizing the dialogs in TwNg, based on current screen resolution (in pixels).

Window Handles and Views

G2 provides several API procedures and corresponding hidden attributes on a g2-window for the management of various types of native view and panes in a Windows environment, such as Telewindows. The categories of native views are:

- MDI child views:
  - Chart views
  - HTML view
  - Dialog views
  - Workspace views
- Panes:
  - Tree view
  - Shortcut bar
  - Property grid

Note Most views are only supported in Telewindows Next Generation (`twng.exe`).

These procedures and attributes allow you to access information about the view, including its coordinates and dimensions in the window, its state, for example, maximized or minimized, its type, and in the case of workspace views, its name.

The callback for all views can receive various event types.

In addition, this category includes system procedures for setting the UI theme and setting tabbed MDI mode.

G2-Window Hidden Attributes

A g2-window that supports Windows views defines two new attributes:

- selected-window-handle — A handle to the selected MDI child view in the window.
- window-handles — A sequence of handles to all MDI child views in the window.

Views API Procedures

The following system procedures allow you to access views from a Telewindows window and to get information about the view.

g2-ui-get-selected-window-handle
(*window*: class g2-window)
-> *handle*: integer

Given a g2-window that supports native views, returns a handle to the selected native view in the window. This system procedure is equivalent to the selected-window-handle attribute of a g2-window that supports native views.

g2-ui-get-window-handles
(*window*: class g2-window)
-> *handles*: sequence

For a g2-window that supports native views, returns a sequence of handles to all native views in the window. This system procedure is the equivalent to the window-handles attribute of a g2-window that supports native views.

g2-ui-is-valid-window-handle
(*handle*: value, *window*: class g2-window)
-> *return-value*: truth-value

For a g2-window that supports native views, returns true if the specified *handle* is a native view, where *handle* is one of the following view designations:

- An integer, which is interpreted as the unique handle of a view as returned by g2-ui-get-selected-window-handle or g2-ui-get-window-handles.
- A symbol is interpreted as the name or class of a view. All views on the window are searched, and the one found whose name matches or the one found whose class matches is used.
- A text, which is interpreted as the title of a view, and the view with that exact title text is used.

If more than one view matches the symbol or text view designation, then the chosen view is undefined.

g2-ui-lookup-window-handle
(*handle*: value, *window*: class g2-window)
-> *info*: structure, *return-value*: truth-value

For a g2-window that supports native views, returns a structure that provides information about the native view specified by *handle*, which is a view designation. For a description of a view designation, see g2-ui-is-valid-window-handle.

The procedure returns true if the specified *handle* is a native view.

The *info* structure provides the following attributes, depending on the type of view. If the handle is invalid, the return structure is empty.

- All views:
  - callback: *symbol* or none — The name of the procedure to call on events.
  - class: *symbol* — The type of native view, which is one of these types: g2-tree-view, g2-html-view, g2-dialog-view, g2-chart-view, g2-shortcut-bar, and g2-workspace-view.

    Note: These are not currently supported as classes.
  - closeable: *truth-value* — Whether the close button is present and enabled.
  - container: *symbol* — The type of native window containing the view. This value affects which attributes and operations apply to the view. Possible values are:
    - mdi-child — The view is in a MDI Child window inside the main Telewindows window.
    - top-level — The view is in a top-level window on the Windows desktop.
    - pane — The view is in a docking pane.
    - embedded — The view is inside the window of another application.
  - handle: *integer* — The integer handle of the native view.
  - height: *integer* — The height of the visible area of the native view in the client window, in pixels. For workspace views, the height may be smaller or larger than the scaled workspace.
  - icon: *item-or-value* — The window icon, which appears at far left of the title bar.
  - left: *integer* — The coordinate of the left side of the visible part of the native view, in pixels, where the coordinate value increases to the right.
  - name: *text* — The name of the workspace, for workspace views, and not-yet-implemented otherwise.
  - state: *symbol* — The current state of the view, which is one of these symbols: normal, minimized, maximized, docked, hidden, autohidden, autoshown, or closed.

Note: The attribute state used to have the value pinned for panes that were autohidden. It now has the value autohidden.

- title: *text* — The text in the title bar.
- top: *integer* — The coordinate of the top of the visible part of the native view, in pixels, where the coordinate value increases downward.
- visible-caption: *truth-value* — Whether the title bar is visible.
- width: *integer*: The width of the visible area of the native view in the client window, in pixels. For workspace views, the width may be smaller or larger than the scaled workspace.

- mdi-child and top-level containers only:
  - minimizeable: *truth-value* — Whether the minimize button is present and enabled.
  - maximizeable: *truth-value*— Whether the maximize button is present and enabled.
  - resizeable: *truth-value* — Whether the view can be resized.
- pane containers only:
  - autohideable: *truth-value* — Whether the pane can be autohidden. When false, the pin is removed from the pane.
  - close-action: *symbol* — Whether to hide or destroy the pane when it is closed. The default is destroy.
  - dock: *symbol* — Where the pane is docked. The options are: left, top, right, bottom, or none (no value).
  - draggable: *truth-value* — Whether the pane can be dragged.
  - floatable: *truth-value* — Whether the pane can be floating.
- g2-workspace-view only:
  - x-scale: *integer* — The vertical scale, where 1.0 is full scale.
  - y-scale: *integer* — The horizontal scale, where 1.0 is full scale.
  - x-scroll-position: *integer* — The vertical scroll position, in pixels.
  - y-scroll-position: *integer* — The horizontal scroll position, in pixels.

g2-ui-manage-pane
(*action*: symbol, *handle*: value, *arg*: value, *win*: g2-window)

Performs an action on the view specified by *handle*, which is a view designation. For a description of a view designation, see g2-ui-is-valid-window-handle. The options for *action* are:

- set-closeable — Adds or removes the close box in a pane's title bar, where *arg* is a truth-value.
- set-title — Specifies the title for the view, where *arg* is the new title as a text.

g2-ui-modify-view
(*handle*: value, *properties*: structure, *window*: class g2-window)

Modifies the view specified by *handle*. The *options* structure may contain any of the attributes in the *info* structure returned by the g2-ui-lookup-window-handle system procedure.

You can also provide attributes when creating a view in the *options* argument of the g2-ui-create-tree-view, g2-ui-create-shortcut-bar, g2-ui-create-chart-view, and g2-ui-create-html-view system procedures.

View Events

In addition to the specific event types, the callback for any type of view can respond to the following general view events:

- autohidden — A docked pane has slid under the main window so that only a tab is now visible.
- autoshown — A docked pane has slid back from under the main window to become completely visible, due to the user hovering the mouse over the pane's tab.
- attached — A pane has been docked within another docking pane.

Workspace Views

You can use the following system procedure to create a workspace view, specifying various information about the view when it is created, such as its position, scale, and scroll position.

To register a callback for a workspace view, use g2-ui-register-workspace-callback described in Workspace Callbacks.

g2-ui-create-workspace-view
(*workspace*: class kb-workspace, *options*: structure, *win*: class g2-window)
-> *handle*: integer

Creates a view of *workspace* on a given window, destroying any other view of it on the same window. The *options* structure permits all the same options as

the return value of g2-ui-lookup-window-handle, which you pass as the *options* argument to this procedure. For details, see the description of g2-ui-lookup-window-handle in Views API Procedures.

In the *options* structure, the only allowable option for container is mdi-child; top-level, pane, and embedded are not allowed.

Themes

g2-ui-set-theme
(*theme*: value, *window*: class g2-window)
-> *theme:* value

Sets the current theme. The options for *theme* are: 2000, 2001, 2002, and 2003, which approximate the appearance of the corresponding release of Microsoft Office. The default theme is 2001. The procedure returns the previous theme.

g2-ui-get-theme
(*window*: class g2-window)
-> *theme:* value

Returns the current theme.

Tabbed MDI Mode

g2-ui-tabbed-mdi-mode
(*tabbed*: truth-value, *options*: structure, *window*: class g2-window)

Switches the user interface between a normal and a tabbed MDI mode. In tabbed MDI mode, when you show a workspace, either interactively or programmatically, the workspace appears as a tab page in the overall window and fills the window. Workspaces include any workspace that appears in its own window, such as KB workspaces, attribute tables, and the G2 Text Editor. For KB workspaces, the tab label corresponds to the name of the workspace. For attribute tables, the tab label indicates the item name and class. For Text Editor workspaces, the tab label includes the words “Text Editor for" followed by the name of the item or attribute being edited. You can interactively switch the order of the tabs by dragging, scroll through the tabs by clicking left and right arrows, and close individual workspaces.

Currently, the *options* structure does not have any options.

Tabbed MDI windows are only supported in Telewindows Next Generation (`twng.exe`).

For examples, see Using Tabbed MDI Mode in Windows Views, Panes, and UI Features in the *G2 Reference Manual*.

35

Version Control

Describes version control procedures for change log entries and text "diff" operations.

Introduction

Use the version control system procedures for:

- Change log entries.
- Text "diff" operations.

Change Log Entry Procedures

Use the following system procedures for programmatically accessing the change log entry for an item.

g2-get-change-log-entry
(*item*: class item, *attribute-name*: symbol, *identifier*: structure)
-> *change-log-entry*: structure

Gets the change log entry for the *attribute-name* attribute of *item*. Use *identifier* to specify a revision, timestamp, or tag to further identify the change log entry to get, which is a structure with this syntax:

structure
(revision: *integer*,
timestamp: *structure*, {see *change-log* return value structure for syntax}
tag: *symbol*)

If *identifier* is empty, it gets the current revision. The structure should specify only one of these attributes.

The *change-log-entry* structure has this syntax:

structure
(attribute: *symbol*,
revision: *integer*,
comment: *text*,
tags: *sequence*, {a sequence of symbolic tags}
text-value: *text*,
module-version: *integer*,
timestamp: structure
(year: *integer*,
month: *integer*,
date: *integer*,
hours: *integer*,
minutes: *integer*),
author: *symbol*)

Note that you cannot use a reserved word as a symbol in the tags sequence or the symbol none, which is not a reserved word.

g2-tag-change-log-entry
(*item*: class item, *attribute-name*: symbol, *identifier*: structure, *tag*: symbol)
-> *change-log*: structure

Adds a tag to the change log entry for the *attribute-name* attribute of *item*. Use *identifier* to specify a revision or timestamp to further identify the change log entry to tag, which is a structure with this syntax:

structure
(revision: *integer*,
timestamp: *structure*)

If *identifier* is empty, it tags the current revision. The structure should specify only one of these attributes.

The *change-log* structure has the same syntax as g2-get-change-log-entry or an empty structure if the tagging failed.

g2-delete-change-log-tag
(*item*: class item, *attribute-name*: symbol, *tag*: symbol)
-> *change-log*: structure

Deletes the value of the specified *tag* for the *attribute-name* attribute of *item*.

The *change-log* structure is the value of the change log entry after the tag has been deleted. It has the same syntax as g2-get-change-log-entry or an empty structure if the deletion failed.

g2-delete-change-log-entry
(*item*: class item, *attribute-name*: symbol, *identifier*: structure)
-> *result*: truth-value

Deletes a change log entry for the *attribute-name* attribute of *item* and returns true if the deletion was successful, false otherwise. Use *identifier* to specify a revision, timestamp, or tag to further identify the change log entry. See g2-get-change-log-entry for a description.

g2-set-change-log-entry-comment
(*item*: class item, *attribute-name*: symbol, *identifier*: structure, *comment*: text)

Sets the value of the comment attribute of the change log entry for the specified *attribute-name* attribute of *item*. Use *identifier* to specify a revision, timestamp, or tag to further identify the change log entry. See g2-get-change-log-entry for a description.

g2-get-change-log-entry-comment
(*item*: class item, *attribute-name*: symbol, *identifier*: structure)
-> *comment*: text

Returns the value of the comment attribute of the change log entry for the specified *attribute-name* attribute of *item*. Use *identifier* to specify a revision, timestamp, or tag to further identify the change log entry. See g2-get-change-log-entry for a description.

g2-revert-change-log-entry
(*item*: class item, *attribute-name*: symbol, *identifier*: structure)
-> *change-log*: structure

Reverts the value of the *slot-name* attribute of *item* to the particular previous value specified by *identifier*. Use *identifier* to specify a revision, timestamp, or

tag to further identify the change log entry. See g2-get-change-log-entry for a description.

The *change-log* structure has the same syntax as g2-get-change-log-entry or an empty structure if the revert failed.

g2-tag-module
(*module-name*: symbol, *identifier*: structure, *tag*: symbol)
-> *result*: truth-value

Sets the value of the tags attribute of the change log entry for every attribute that has a change log of every item in the specified *module-name* with *tag*, and returns true if the tagging was successful, false otherwise. If the tags attribute already has a value, this system procedure adds to the existing sequence. For example, if the value of the tags attribute is sequence(abc), calling g2-tag-module with xyz as the *tag* results in sequence(abc, xyz). Use *identifier* to specify a timestamp to further identify the change log entry. If *identifier* is empty, it tags the current revision. See g2-tag-change-log-entry for a description.

g2-revert-module
(*module-name*: symbol, *identifier*: structure)
-> *result*: truth-value

Reverts the value of the *attribute-name* attribute of *item* to the particular previous value specified by *identifier*. Use *identifier* to specify a revision, timestamp, or tag to further identify the change log entry. See g2-get-change-log-entry for a description.

g2-disable-change-logging-on-item
(*item*: class item)

Disables change logging on the specified *item* in a module. If change logging is already disabled on the module or on the item, this system procedure has no effect.

g2-enable-change-logging-on-item
(*item*: class item)

Enables change logging on the specified *item* in a module. If change logging is disabled on the module, this system procedure generates an error. If change logging is already enabled on the item, this system procedure has no effect.

Text “Diff” Procedures

Use the following system procedures to perform a “diff” operation on two texts and change log entries. You specify an external program to perform the “diff” operations; G2 does not provide an internal capability for performing “diff” operations.

g2-set-external-diff-specification
(*spec*: structure)
-> *result*: truth-value

Change the command used for performing the “diff” operation when using g2-diff-texts. The *spec* structure has this syntax:

structure
(pathname: *text*,
command-line-options: *text*)

where:

- pathname is the path to the “diff” program to use.
- command-line-options is a text string of command-line options for the “diff” program.

g2-diff-texts
(*text1:* text, *text2*: text)
-> *result*: structure

Returns a structure that describes the difference between two texts. The syntax for *result* is:

structure
(diff-output: *text*,
diff-error: *text*)

where:

- diff-output describes the differences in the two input text or an empty text, if the diff fails.
- diff-error is the empty text if the “diff” succeeds, or the error text.

By default, the system procedure uses `/usr/bin/diff` on UNIX and `C:\WINDOWS\system32\fc.exe` on Windows. To change the “diff” command that g2-diff-texts uses, use g2-set-external-diff-specification.

g2-diff-change-log-entries
(*item*: class item, *attribute-name*: symbol, *identifier-1*: structure, *identifier-2*: structure)
-> *result*: structure

Returns a structure that describes the difference between two change log entries, where *change-log-spec-1* and *change-log-spec-2* are of the format returned by g2-get-change-log-entry. The change log specifications do not have to be of the same type, for example, one could be a timestamp and the other could be a tag.

This system procedure uses the same "diff" programs as g2-diff-texts.

The syntax for *result* is the same as for g2-diff-texts.

36

Version Information Operations

Describes the procedures for obtaining version information for your G2 process and your sys-mod KB.

Introduction

Use the version information system procedures to access G2 version information as a text or structure value, and to obtain the version of sys-mod you are using.

g2-get-g2-version-information

Returns G2 version information in a structure value.

Synopsis

g2-get-g2-version-information
( )
-> *return-value*: structure

Return Value	Description
return-value	The returned structure has these attribute names and value types: major-version-number: integer minor-version-number: integer revision: integer release-level: symbol { alpha | beta | release | prototype } patch-number: integer or false build-identification: text build-comment: text or false

Example

Here is an example return value:

```
structure(
    MAJOR-VERSION-NUMBER: 6,
    MINOR-VERSION-NUMBER: 0,
    REVISION: 2,
    RELEASE-LEVEL: the symbol ALPHA,
    PATCH-NUMBER: false,
    BUILD-IDENTIFICATION: "CD11"
    BUILD-COMMENT: false)
```

g2-get-g2-version-of-kb-file

Gets the G2 version of a KB file without having to load the KB.

Synopsis

g2-get-g2-version-of-kb-file
(*file-string*: text, *post-flag*: truth-value)
-> *version*: structure

Arguments	Description
file-string	A text string that is the complete path name to a KB file.
post-flag	If true, prints the version text to the Message Board.

Return Value	Description
version	A structure that describes the KB version.

Description

The syntax of the return structure is:

structure
(version-text: *text*,
major-version-number: *integer*,
minor-version-number: *integer*,
revision-text: *text*)

For example:

structure
(version-text: "G2 8.1 Rev. 0",
major-version-number: 8,
minor-version-number: 1,
revision-text: "Rev. 0")

In case of an error, the procedure returns structure(version-text: ""). A message describing the error also appears either in the Message Board or the Operator Logbook.

g2-get-software-version

Returns the current version of the G2 software as a text value.

Synopsis

g2-get-software-version
( )
-> *g2-version*: text

Return Value	Description
g2-version	A text value of the system and software version of the current G2 process.

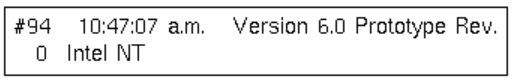
Description

This system procedure does not require any arguments. It returns a text value of the current G2 system and software version.

On HP workstations, the system version always returns HPUX, regardless of the operating system version.

Examples

A procedures that gets the version information about the current G2 and posts it on the Message Board, is:




```
get-g2-version( )
version-text: text;
begin
    version-text = call g2-get-software-version( );
    post "[version-text]"
end
```

g2-get-sys-mod-version

Returns the version number of the currently loaded sys-mod module.

Synopsis

g2-get-sys-mod-version
( )
-> *version*: float, *revision*: integer, *type*: symbol

Return Value	Description
version	The float major and minor version number of the sys-mod module, such as 7.0.
revision	The integer revision of the sys-mod module, such as 0, 1, and so on.
type	The symbol type release of the sys-mod module, either prototype, alpha, beta, or release.

Description

This system procedure provides version information about the current sys-mod module.

Example

This procedure posts the current sys-mod version:

```
post-sys-mod-version( )
version: float;
revision: integer;
version-type: symbol;
begin
    version, revision, version-type = call g2-get-sys-mod-version( );
    post "The current version of SYS-MOD is [version] Rev. [revision],
        and it is a [version-type] release."
end
```

37

Web Operations

Describes the procedures for networking and integration with Web services, SOAP, and HTTP.

Introduction

Use the web operation system procedures for networking and integration with Web services, SOAP, and HTTP.

For related system procedures used for interfacing with TCP/IP sockets, see Network Connection Management and Network Reading and Writing.

The system procedures described in this chapter are located in `g2web.kb`, which is in `g2\kbs\utils` or `g2/kbs/utils`, depending on your platform.

For examples of interfacing with Web Services, SOAP, and HTTP, see `g2web-demo.kb`, located in the `g2\kbs\demos` or `g2/kbs/demos` directory of your G2 Bundle installation directory.

Web Services

According to the World Wide Web Consortium (W3C) Web Services Architecture Working Group Note (`http://www.w3.org/TR/ws-arch/`):

> A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

G2 can act as a Web service requester agent (client), using the following system procedures in `g2web.kb`.

g2-import-web-service-description
(*url*: text)
-> *description*: class web-service-description

Imports a Web service description from a URL on the Web. This system procedure is equivalent to using g2-send-web-request and importing it by using g2-import-web-service-description-from-xml-text.

g2-import-web-service-description-from-xml-text
(*document*: text)
-> *description*: class web-service-description

Returns a web-service-description item from a WSDL *document*, expressed as XML text. Items representing the interfaces, bindings, and services defined by the document are placed on the subworkspace of the returned *description* item. An error is signaled if the text argument is not a well-formed WSDL document.

Currently, only WSDL 1.1 documents are supported. However, the names for G2 Web service system classes and attributes use the terminology of WSDL 2.0. In WSDL 1.1 terminology, an endpoint is a port and an interface is a `portType`. WSDL 1.1 messages do not exist in WSDL 2.0; an interface refers to schema elements directly through its interface message references.

g2-invoke-web-service-operation
(*endpoint-reference*: structure, *operation-name*: symbol,
input-message: structure)
-> *output-message*: structure

Invokes an operation on a remote Web service given an *endpoint-reference* structure and *operation-name*, and returns the *output-message*. The *endpoint-reference* is a structure with this syntax:

structure
(service-namespace: *text*,

service-name: *text*,
endpoint-name: *text*)

The service-name and endpoint-name attributes identify an endpoint according to the web-service-description item whose XML namespace matches the service-namespace attribute.

An error is signaled if the *endpoint-reference* does not identify a valid endpoint, if the input message is not well-formed, or if an error occurs while communicating with the Web service.

, if the input message is not well-formed, or if an error occurs while communicating with the Web service.

The *input-message* and *output-message* are Web service message structures, which is a structure whose attributes correspond to message parts. The value of a message part attribute is a text, an XML element value, or a sequence of texts and/or XML element values.

An XML element value is a structure representing an XML element with this syntax:

structure
(tag-name: *text*,
attributes: *structure*,
children: *sequence*)

where:

- tag-name is the element tag name. This attribute is required.
- attributes is a structure containing named attribute values, which are texts. This attribute is optional.
- children is a sequence of XML elements and/or texts. This attribute is optional.

Attribute names use the same correspondence between XML names and G2 symbols used by G2GL, for example, `myAttribute` becomes my-attribute.

For example, this XML text:

```
<elt attrName="attrValue">
    <child>text1</child>
    text2
</elt>
```

corresponds to this XML element value:

```
structure
    (tag-name: "elt",
    attributes: structure
        (attr-name: "attrValue"),
        children: sequence (structure
```

```
(tag-name: "child",
children: sequence ("text1")), "text2"))
```

SOAP

G2 can send and receive SOAP 1.1 requests, using the following system procedures in `g2web.kb`.

g2-send-soap-request
(*url*: text, *request*: structure)
-> *response*: structure

Sends a SOAP 1.1 request to a SOAP receiver at the given URL, returning the SOAP response when it arrives. Currently, only HTTP URLs are supported.

The *request* structure has the following syntax:

```
structure
(header-entries: sequence,
 body-entries: sequence,
 action: text)
```

where:

- header-entries — A sequence of XML element values. For a description of XML element values, see g2-invoke-web-service-operation.
- body-entries — A sequence of XML element values. This is the only required attribute of the structure.
- action — The `SOAPAction` URI, indicating the intent of the request; this attribute is only used for HTTP.

The *response* structure has the following syntax:

```
structure
(header-entries: sequence,
body-entries: sequence)
```

g2-handle-http-request-as-soap
(*server*: class http-server, *http-request*: structure,
soap-dispatch: class procedure, *user-data*: item-or-value)
-> *response*: structure

Converts an *http-request* message into a SOAP request structure, passes it to the *soap-dispatch* procedure, and converts the resulting SOAP *response* structure into an HTTP response. The g2-handle-http-request-as-soap system procedure is intended to be called by the http-server-dispatch procedure of an http-server, which is described under g2-start-http-server. You can configure *user-data* to be any value, for example, you might want to pre-compute session information in the HTTP handler before determining if the request is a SOAP request, then pass the pre-computed information to the SOAP handler.

HTTP

Web Client

G2 can act as a Web client, using the following system procedure in `g2web.kb`.

g2-send-web-request
(*url*: text, *request*: structure)
-> *response*: structure

Sends a request to a Web server at the given *url*, returning the response when it arrives. The *request* structure and the returned *response* structure depend on the URL scheme. Currently, only HTTP URLs are supported. The *request* structure has the following syntax, where all attributes are optional:

structure
(method: *symbol*,
headers: *structure*,
entity: *structure* | *text*)

where:

- method — The HTTP method of the request, such as get or post. The default is get.
- headers — A structure of HTTP header attributes included in the request.
- entity — If provided as a structure, the attributes of the structure are encoded using the application/x-www-form-urlencoded MIME type. If provided as a text, the body of the request, which is assumed to already be encoded into 8-bit characters. Note that providing the entity as a text is only valid if the method is post.

The *response* structure has the following syntax:

structure
(http-version: *text*,
status-code: *integer*
reason-phrase: *text*,
headers: *structure*,
transfer-length: *integer*,
connection: *g2-socket*)

where:

- http-version — The HTTP version, for example, "HTTP/1.1".
- status-code — The RFC 2616 status code of the response message.
- reason-phrase — The reason phrase in the HTTP response message.
- headers — The message headers from the HTTP response message.

- transfer-length — The number of bytes to be transferred as part of the entity. If the transfer-length is -1, the entity length is undetermined and all bytes should be read from the connection.
- connection — If transfer-length is nonzero, a g2-socket instance from which the response entity can be read.

If the transfer-length is nonzero, the caller of the Web request is responsible for closing the g2-socket connection by using the g2-tcp-close system procedure.

g2-read-http-entity-body
(*http-message*: structure)
-> *entity-body*: text

Reads the *http-message*, decoding it according to the Transfer-Encoding header, as required, and returns the entity-body of an HTTP message as a text. The *http-message* structure is the same as the *response* structure returned by g2-send-web-request.

g2-check-http-response-status-code
(*response*: structure)

Examines an HTTP *response* structure and signals an error if the status code indicates an HTTP error.

HTTP Server

G2 can act as an HTTP server, using the following system procedures in `g2web.kb`.

g2-start-http-server
(*server*: class http-server, *port*: integer)

Starts a task that listens on the given TCP/IP *port* number for HTTP 1.1 requests and passes them to the dispatch procedure of the specified *server*. The dispatch procedure is the value of the http-server-dispatch attribute of the http-server instance. The http-server-port attribute of the *server* is set to the specified *port*. If the server is already currently listening, it is shut down first.

The http-server-dispatch attribute of an http-server item names a G2 procedure with the following signature:

my-http-server-dispatch-procedure
(*server*: class http-server, *request*: structure)
-> *response*: structure

For each HTTP request that is received by the server, this procedure is called with the *server* and a *request* structure, which has the following syntax:

structure
(method: *symbol*,
request-uri: *text*,
http-version: *text*,
path: *sequence*,
query: *text*,
headers: *structure*,
entity: *text*)

where:

- method — The HTTP method of the request, such as get or post.
- request-uri — The URI of the request, which is everything after the host name in a URL. For example, if the URL is "http://www.gensym.com/index.asp?p=gensym_in_the_news", the URI is "/index.asp?p=gensym_in_the_news".
- http-version — The HTTP version, for example, "HTTP/1.1".
- path —A sequence of path segments in the request-uri. The path sequence begins with the first segment after the leading slash. In the example above, the path would be sequence("index.asp"). If there is no path, that is, the request-uri is an absolute URI with no slashes after the host, then the path sequence is empty. If the path is just "/", then the path sequence contains one empty text.
- query — (Optional) The portion of the URI that is the HTTP query, which follows the ?. In the example above, the query would be "p=gensym_in_the_news".
- headers — A structure of HTTP header attributes included in the request.
- entity — (Optional) The entity of the request, if any.

The *response* structure has the following syntax, where all the attributes are optional and no other attributes can be included:

structure
(status-code: *integer*,
reason-phrase: *text*,
headers: *structure*,
entity: *text*,
entity-producer: *procedure*,
user-data: *item-or-value*)

where:

- status-code — The RFC 2616 status code of the response. The default is 200 (OK).

- reason-phrase — A custom reason phrase. The default is the recommended reason phrase from RFC 2616 for the given status-code.
- headers — A structure of HTTP header attributes to be included in the request.
- entity — The entire entity text to be included in the response.
- entity-producer — A procedure that returns the entity incrementally. This attribute should be present when the response contains an entity that is too large to fit into a single entity text. The signature of the procedure is:

  my-entity-producer
  (*response*: structure)
  -> *entity-bytes*: sequence

  After the response headers are written to the HTTP connection, if the response structure includes an entity-producer attribute, the procedure that is its value is called repeatedly, and each return value sequence is written to the HTTP connection, using g2-tcp-write-bytes. An empty sequence signals the end of the entity.
- user-data — Any user-defined item or value. This can be used to pass other data to the entity-producer procedure.

The user may change the http-server-dispatch attribute while a server is running, in which case each incoming request goes to the current dispatch procedure. If this attribute is none or does not name an existing procedure, all requests receive a generic "404 not found" response.

g2-shutdown-http-server
(*server*: class http-server)

Stops the listener task of the specified *server*. The http-server-port attribute is set to NONE. This procedure does nothing if the server is not currently listening.

38

XML Parser API

Describes procedures for parsing XML files.

Introduction

G2 provides a way of parsing XML code and executing callbacks. G2 uses SAX (Simple API for XML), an event-based XML parser that is one of several standards for parsing XML code.

The G2 XML parser consists of a parser class and a set of G2 system procedures that parse the XML text and execute callbacks. First, you must convert the XML code to text. Once the code is in text format, you write a procedure that parses the text, then sequentially executes callbacks. The callback can be any user-defined procedure with a given signature.

You can associate callbacks with the start and end of the XML document, the start and end of an XML element, and non-XML characters in the document. For performance reasons, you typically parse the XML text in "chunks." When parsing is complete, you must dispose of the parser object.

For more information, see XML Parsing in the *G2 Reference Manual*.

SAX Parser API

g2-sax-parse-text
(*sax-parser*: class sax-parser, *text-input*: text)
-> *callbacks-ready*: integer

Argument	Description
sax-parser	The sax-parser instance that contains the XML text to parse.
text-input	The XML code to parse, as a text. The procedure parses the entire *text-input* at once. You provide the XML code as text. For details, see XML Parsing in the *G2 Reference Manual*.

Return Value	Description
callbacks-ready	The number of callbacks remaining to execute, which is equivalent to the number-of-pending-callbacks attribute of the sax-parser object.

If the length of the *text-input* is small, you can use this procedure to parse the entire *text-input* at once. However, keep in mind that no other processing can occur while G2 parses the text. If the length of the text input is large, use g2-sax-parse-chunk instead to parse the text in smaller chunks.

For example:

```
my-sax-parser: class sax-parser;
text-input: text;
callbacks-ready: integer;

callbacks-ready = call g2-sax-parse-text(my-sax-parser, text-input);
```

g2-sax-parse-chunk
(*sax-parser*: class sax-parser, *text-input*: text,
start-index: integer, *end-index*: integer)
-> *callbacks-ready*: integer

Argument	Description
sax-parser	The sax-parser instance that will parse the XML text.
text-input	The XML code to parse, as a text. The procedure parses the text from *start-index* to *end-index*. You provide the XML code as text. For details, see XML Parsing in the *G2 Reference Manual*.
start-index	The index of *text-input* where parsing should start, as an integer.
end-index	The index of *text-input* where parsing should stop, as an integer.

Return Value	Description
callbacks-ready	The number of callbacks remaining to execute, which is equivalent to the number-of-pending-callbacks attribute of the sax-parser object.

If the length of the text input is large, you can use this procedure to parse the *text-input* in chunks. By parsing the text in chunks, G2 can execute callbacks and perform other processing more frequently.

The size of the "chunk" of text to parse depends on the length of the text input, the number of callbacks, and your processor speed. Experiment with different sizes to fine-tune the performance.

The *start-index* and *end-index* use the same mechanism that the G2 text functions use, where the first position is index 1, and the last position is inclusive. For details, see XML Parsing in the *G2 Reference Manual*.

For example:

```
my-sax-parser: class sax-parser;
text-input: text;
start-index, end-index, callbacks-ready: integer;

callbacks-ready = call g2-sax-parse-chunk(my-sax-parser, text-input, start-index,
end-index);
```

g2-sax-execute-next-callback
(*sax-parser*: class sax-parser)

Argument	Description
sax-parser	The sax-parser instance whose next callback should be executed.

Use this procedure to execute each pending callback in the number-of-pending-callbacks attribute of the sax-parser object. When g2-parse-text or g2-parse-chunk is called on the input text string, G2 queues the callbacks to execute, based on the events it encounters, for example, start document, end document, start element, end element, and so on. When parsing is complete, calling g2-sax-execute-next-callback executes the user-defined callback procedure associated with the oldest callback in the queue. It then removes that callback from the queue and decrements number-of-pending-callbacks by one.

For example:

```
my-sax-parser: class sax-parser;

call g2-sax-execute-next-callback (my-sax-parser)
```

g2-sax-finish-parsing
(*sax-parser*: class sax-parser)

Argument	Description
sax-parser	The sax-parser instance whose parsing is complete.

Use this procedure to signal that parsing is complete. The callback procedure associated with the end-document-procedure can execute only after you call g2-sax-finish-parsing. You call this procedure after you call g2-sax-parse-text or g2-sax-parse-chunk.

For example:

```
my-sax-parser: class sax-parser;

call g2-sax-finish-parsing(my-sax-parser)
```

g2-sax-reset-parser
(*sax-parser*: class sax-parser)

Argument	Description
sax-parser	The sax-parser instance to reset.

Use this procedure to reset the sax-parser instance. Resetting the parser deletes all pending callbacks. You might want to call this procedure if you are going to reuse the parser or if a parsing error occurs.

For example:

```
my-sax-parser: class sax-parser;

call g2-sax-reset-parser(my-sax-parser)
```

g2-sax-parse-file
(*sax-parser*: class sax-parser, *file-input*: text)
-> *callbacks-ready*: integer

Argument	Description
sax-parser	The sax-parser instance that will parse the XML document.
file-input	A path to an XML file to parse, as a text. The procedure parses the entire *file-input* at once. For details, see XML Parsing in the *G2 Reference Manual*.

Return Value	Description
callbacks-ready	The number of callbacks remaining to execute, which is equivalent to the number-of-pending-callbacks attribute of the sax-parser object.

Although this approach takes up more memory than reading the file in chunks, it avoids the limitation on the maximum size of a text and is easier than parsing the file in chunks. Note that this system procedure does not allow other processing while reading the file, which might be an issue for very large files or slow file systems.

A

Obtaining the G2-Window of an Item

Describes techniques for mapping an item to its G2-window.

Introduction

Many G2 system procedures include an argument specifying the current g2-window. Currently, there is no expression in G2 such as:

```
the g2-window of my-item
```

To access a particular g2-window, using on of the following two statements:

- this window
- if there exists a g2-window

Using the This Window Statement

You can use the this window statement in either an action button or a user-menu-choice that starts a procedure. These are the only items in G2 that support this statement, since both items must be visible on some window to be chosen.

The example for the system procedure g2-get-workspace-layer-position shows how to pass a window to a procedure from within an action button.

Using Existence Statements

You can use the if there exists a g2-window statement from within a method or procedure to pass a g2-window to the system procedure, as follows:

```
if there exists a g2-window G then
    call g2-system-procedure (G)
else
    inform the operator that "No g2-window exists!";
```

For a rule, use the syntax above, but without the else statement.

B

Chart Properties and Enumeration Values

Describes the known chart view properties and enumeration values.

Introduction

This appendix lists the known chart view properties and their types, as well as known chart view enumeration values.

For detailed documentation on these properties, see `http://gigasoft.com/WebHelp/Peonlref.htm`.

Known Chart View Properties

Notes:

- The type *color* means a symbol naming a G2 color or of the form *RGBxxyyzz*.
- The type *sequence*(*float*) means a sequence containing only floats.

3d-dialogs: *truth-value*
3d-threshold: *integer*
add-skirts: *truth-value*
allow-annotation-control: *truth-value*
allow-area: *truth-value*
allow-axis-hot-spots: *truth-value*

allow-axis-label-hot-spots: *truth-value*
allow-axis-page: *truth-value*
allow-bar: *truth-value*
allow-best-fit-curve: *truth-value*
allow-best-fit-curve-ii: *truth-value*
allow-best-fit-line: *truth-value*
allow-best-fit-line-ii: *truth-value*
allow-bottom-title-hot-spots: *truth-value*
allow-bubble: *truth-value*
allow-color-page: *truth-value*
allow-contour-colors: *truth-value*
allow-contour-control: *truth-value*
allow-contour-lines: *truth-value*
allow-coord-prompting: *truth-value*
allow-customization: *truth-value*
allow-data-hot-spots: *truth-value*
allow-data-labels: *integer*
allow-debug-output: *truth-value*
allow-exporting: *truth-value*
allow-font-page: *truth-value*
allow-graph-annot-hot-spots: *truth-value*
allow-graph-hot-spots: *truth-value*
allow-grid-number-hot-spots-x: *truth-value*
allow-grid-number-hot-spots-y: *truth-value*
allow-grid-number-hot-spots-z: *truth-value*
allow-histogram: *truth-value*
allow-horizontal-bar: *truth-value*
allow-horz-bar-stacked: *truth-value*
allow-horz-line-annot-hot-spots: *truth-value*
allow-horz-scroll-bar: *truth-value*
allow-jpeg-output: *truth-value*
allow-line: *truth-value*
allow-maximization: *truth-value*
allow-ole-export: *truth-value*
allow-page1: *truth-value*
allow-page2: *truth-value*
allow-plot-customization: *truth-value*
allow-point: *truth-value*
allow-point-hot-spots: *truth-value*
allow-points-page: *truth-value*
allow-points-plus-line: *truth-value*
allow-points-plus-spline: *truth-value*
allow-popup: *truth-value*
allow-ribbon: *truth-value*
allow-rotation: *truth-value*
allow-spline: *truth-value*
allow-step: *truth-value*

allow-stick: *truth-value*
allow-style-page: *truth-value*
allow-subset-hot-spots: *truth-value*
allow-subsets-page: *truth-value*
allow-subtitle-hot-spots: *truth-value*
allow-surface: *truth-value*
allow-surface-contour: *truth-value*
allow-surface-pixel: *truth-value*
allow-surface-shading: *truth-value*
allow-table: *truth-value*
allow-table-hot-spots: *truth-value*
allow-title-hot-spots: *truth-value*
allow-titlesdialog: *truth-value*
allow-user-interface: *truth-value*
allow-vert-line-annot-hot-spots: *truth-value*
allow-vert-scroll-bar: *truth-value*
allow-wire-frame: *truth-value*
allow-x-axis-annot-hot-spots: *truth-value*
allow-y-axis-annot-hot-spots: *truth-value*
allow-zooming: *integer*
alt-freq-threshold: *integer*
alt-frequencies: *sequence*(*integer*)
annotation-color: *color*
annotations-in-front: *truth-value*
annotations-on-surfaces: *truth-value*
append-point-colors: *sequence*(*color*)
append-point-label-data: *sequence*(*text*)
append-to-end: *truth-value*
append-with-no-update: *truth-value*
append-xdata: *sequence*(*float*)
append-ydata: *sequence*(*float*)
append-zdata: *sequence*(*float*)
assumes-eqdata: *truth-value*
auto-explode: *integer*
auto-image-reset: *truth-value*
auto-min-max-padding: *integer*
auto-rotation: *truth-value*
auto-scale-data: *truth-value*
auto-stat-subsets: *sequence*(*integer*)
auto-xdata: *integer*
axes-annotation-text-size: *integer*
axis-format-ry: *text*
axis-format-tx: *text*
axis-format-x: *text*
axis-format-y: *text*
axis-location-ry: *integer*
axis-location-y: *integer*

axis-number-spacing: *float*
axis-size-ry: *integer*
axis-size-y: *integer*
axis-tick-spacing: *float*
bar-border-color: color
bar-width: *float*
best-fit-coeffs: *sequence*(*float*)
best-fit-degree: *integer*
best-fit-fix: *truth-value*
bitmap-gradient-menu: *integer*
bitmap-gradient-mode: *truth-value*
border-type-menu: *integer*
border-types: *integer*
bottom-margin: *text*
box-plot-color: *color*
bubble-size: *integer*
cachebmp: *truth-value*
cannotation-color: *color*
cdata-label-type: *integer*
cdata-precision: *integer*
cdata-shadows: *truth-value*
cdesk-color: *color*
cfontsize: *integer*
cforce-vertical-points: *integer*
cgraph-back-color: *color*
cgraph-data-labels: *truth-value*
cgraph-fore-color: *color*
cgraphplustable: *integer*
cgrid-in-front: *truth-value*
cgrid-line-control: *integer*
cgrouping-percent: *integer*
char-set: *integer*
clabel-bold: *truth-value*
clabel-font: *text*
clabel-italic: *truth-value*
clabel-underline: *truth-value*
comparison-subsets: *integer*
contour-labels: *sequence*(*text*)
contour-line-label-density: *integer*
contour-menu: *integer*
contour-style-legend: *truth-value*
control-belongs-to-max-dlg: *truth-value*
cursor-mode: *integer*
cursor-page-amount: *integer*
cursor-point: *integer*
cursor-prompt-location: *integer*
cursor-prompt-style: *integer*

cursor-prompt-tracking: *truth-value*
cursor-subset: *integer*
curve-granularity: *integer*
custom: *truth-value*
custom-grid-numbers *struct*
custom-grid-numbers-ry: *truth-value*
custom-grid-numbers-tx: *truth-value*
custom-grid-numbers-x: *truth-value*
custom-grid-numbers-y: *truth-value*
custom-grid-numbers-z: *truth-value*
custom-menu: *sequence*(*integer*)
custom-menu-location: *sequence*(*integer*)
custom-menu-state: *sequence*(*integer*)
custom-menu-text: *sequence*(*text*)
customize-dialog-menu: *integer*
darktextinset: *truth-value*
dash-linethickness: *float*
data-hot-spot-limit: *integer*
data-label-type: *integer*
data-point-labels: *sequence*(*text*)
data-precision: *integer*
data-precision-menu: *integer*
data-shadow-menu: *integer*
data-shadows: *truth-value*
date-time-mode: *integer*
date-time-show-seconds: *truth-value*
day-label-type: *integer*
day-light-savings: *truth-value*
de-limiter: *integer*
def-orientation: *integer*
degree-of-rotation: *integer*
degree-prompting: *truth-value*
delta-x: *integer*
deltas-per-day: *integer*
desk-bmp-filename: *text*
desk-bmp-style: *integer*
desk-color: *color*
desk-gradient-end: *integer*
desk-gradient-start: *integer*
desk-gradient-style: *integer*
dialog-result: *integer*
dialog-shown: *truth-value*
dirty: *truth-value*
disable-clipping: *truth-value*
disable-3d-shadow: *truth-value*
disable-sort-plot-methods: *truth-value*
disable-symbol-fix: *truth-value*

drop-shadow-off-set-x: *integer*
drop-shadow-off-set-y: *integer*
drop-shadow-steps: *integer*
drop-shadow-width: *integer*
end-time: *float*
eng-station-format: *truth-value*
export-dest-def: *integer*
export-dialog-menu: *integer*
export-file-def: *text*
export-size-def: *integer*
export-type-def: *integer*
export-unit-xdef: *text*
export-unit-ydef: *text*
extra-axis-tx *struct*
extra-axisx *struct*
fall-daylight *struct*
first-pt-label-offset: *integer*
fixed-font-menu: *integer*
fixed-fonts: *truth-value*
fixed-line-thickness: *truth-value*
fixed-spmwidth: *truth-value*
floating-bars: *truth-value*
floating-stacked-bars: *truth-value*
focal-rect: *truth-value*
font-size: *integer*
font-size-alcntl: *float*
font-size-axis-cntl: *float*
font-size-cpcntl: *float*
font-size-global-cntl: *float*
font-size-gncntl: *float*
font-size-legend-cntl: *float*
font-size-mbcntl: *float*
font-size-menu: *integer*
font-size-mscntl: *float*
font-size-tbcntl: *float*
font-size-title-cntl: *float*
force-right-y-axis: *truth-value*
force-top-x-axis: *truth-value*
force-vert-points-menu: *integer*
force-vertical-points: *integer*
format-table: *truth-value*
gradient-bars: *integer*
graph-annot-back-color: *color*
graph-annotation-axis: *sequence*(*integer*)
graph-annotation-color: *sequence*(*color*)
graph-annotation-hot-spot: *sequence*(*integer*)
graph-annotation-text: *sequence*(*text*)

graph-annotation-text-size: *integer*
graph-annotation-type: *sequence*(*integer*)
graph-annotation-x: *sequence*(*float*)
graph-annotation-y: *sequence*(*float*)
graph-annotation-z: *sequence*(*float*)
graph-annotationsizecntl: *float*
graph-annotmoveable: *integer*
graph-annottextdodge: *integer*
graph-annottextlocation: *sequence*(*integer*)
graph-back-color: *color*
graph-bmp-filename: *text*
graph-bmp-style: *integer*
graph-data-labels: *truth-value*
graph-fore-color: *color*
graph-gradient-end: *integer*
graph-gradient-start: *integer*
graph-gradient-style: *integer*
graph-loc *struct*
graph-plus-table: *integer*
graph-plus-table-menu: *integer*
grid-aspect: *float*
grid-hot-spot-value: *sequence*(*float*)
grid-in-front: *truth-value*
grid-in-front-menu: *integer*
grid-line-control: *integer*
grid-line-menu: *integer*
grid-style: *integer*
group-percent-menu: *integer*
grouping-percent: *integer*
hatch-back-color: *color*
help-file-name: *text*
help-menu: *integer*
hide-intersecting-text: *integer*
horz-line-annot-hot-spot: *sequence*(*integer*)
horz-line-annotation: *sequence*(*float*)
horz-line-annotation-axis: *sequence*(*integer*)
horz-line-annotation-color: *sequence*(*color*)
horz-line-annotation-text: *sequence*(*text*)
horz-line-annotation-type: *sequence*(*integer*)
horz-scroll-pos: *integer*
hot-spot-data *struct*
hot-spot-size: *integer*
hour-glass-threshold: *integer*
hscroll-style: *integer*
image-adjust-bottom: *integer*
image-adjust-left: *integer*
image-adjust-right: *integer*

image-adjust-top: *integer*
include-data-labels-menu: *integer*
initial-scale-for-rydata: *integer*
initial-scale-for-txdata: *integer*
initial-scale-for-xdata: *integer*
initial-scale-for-ydata: *integer*
initial-scale-for-zdata: *integer*
invalid: *truth-value*
inverted-ry-axis: *truth-value*
inverted-tx-axis: *truth-value*
inverted-x-axis: *truth-value*
inverted-y-axis: *truth-value*
inverted-z-axis: *truth-value*
jpg-quality: *integer*
keydown-data *struct*
label-bold: *truth-value*
label-font: *text*
label-italic: *truth-value*
label-underline: *truth-value*
last-menu-index: *integer*
last-mouse-move pt
last-sub-menu-index: *integer*
left-edge-spacing: *float*
left-margin: *text*
legend-annotation-axis: *sequence*(*integer*)
legend-annotation-color: *sequence*(*color*)
legend-annotation-text: *sequence*(*text*)
legend-annotation-type: *sequence*(*integer*)
legend-location: *integer*
legend-location-menu: *integer*
legend-style: *integer*
line-annotation-text-size: *integer*
line-gap-threshold: *float*
line-shadows: *truth-value*
log-scale-exp-labels: *truth-value*
log-tick-threshold: *integer*
logical-limit: *integer*
long-x-axis-tick-menu: *integer*
long-y-axis-tick-menu: *integer*
lower-bound-text: *text*
lower-bound-value: *float*
main-title: *text*
main-title-bold: *truth-value*
main-title-font: *text*
main-title-italic: *truth-value*
main-title-underline: *truth-value*
manual-contour-line: *float*

manual-contour-max: *float*
manual-contour-min: *float*
manual-contour-scale-control: *integer*
manual-max-data-string: *text*
manual-max-point-label: *text*
manual-max-ry: *float*
manual-max-tx: *float*
manual-max-x: *float*
manual-max-y: *float*
manual-max-z: *float*
manual-min-ry: *float*
manual-min-tx: *float*
manual-min-x: *float*
manual-min-y: *float*
manual-min-z: *float*
manual-ry-axis-line: *float*
manual-ry-axis-tick: *float*
manual-ry-axis-tickn-line: *truth-value*
manual-scale-control-ry: *integer*
manual-scale-control-tx: *integer*
manual-scale-control-z: *integer*
manual-scale-controlx: *integer*
manual-scale-controly: *integer*
manual-stacked-max-y: *float*
manual-stacked-min-y: *float*
manual-tx-axis-line: *float*
manual-tx-axis-tick: *float*
manual-tx-axis-tickn-line: *truth-value*
manual-x-axis-line: *float*
manual-x-axis-tick: *float*
manual-x-axis-tickn-line: *truth-value*
manual-y-axis-line: *float*
manual-y-axis-tick: *float*
manual-y-axis-tickn-line: *truth-value*
manual-z-axis-line: *float*
manual-z-axis-tick: *float*
manual-z-axis-tickn-line: *truth-value*
manualslice-labellength: *integer*
mark-data-points: *truth-value*
mark-data-points-menu: *integer*
max-axis-annotation-cluster: *integer*
max-data-precision: *integer*
max-points-to-graph: *integer*
maximize-menu: *integer*
minimum-point-size: *integer*
mintable-fontsize: *integer*
modal-dialogs: *truth-value*

modeless-auto-close: *truth-value*
modeless-on-top: *truth-value*
mono-desk-color: *color*
mono-graph-back-color: *color*
mono-graph-fore-color: *color*
mono-shadow-color: *color*
mono-table-back-color: *color*
mono-table-fore-color: *color*
mono-text-color: *color*
mono-with-symbols: *truth-value*
month-label-type: *integer*
mouse-cursor-control: *truth-value*
mouse-wheel-function: *integer*
multi-axes-proportions: *sequence*(*float*)
multi-axes-separators: *integer*
multi-axes-sizing: *integer*
multi-axes-subsets: *sequence*(*integer*)
multi-axis-separator-size: *integer*
multi-axis-style: *integer*
multi-axis-style-menu: *integer*
multi-bottom-titles: *sequence*(*text*)
multi-subtitles: *sequence*(*text*)
negative-from-x-axis: *truth-value*
no-custom-parms: *truth-value*
no-grid-line-multiples: *truth-value*
no-help: *truth-value*
no-hidden-lines-in-area: *truth-value*
no-random-points-to-export: *truth-value*
no-random-points-to-graph: *truth-value*
no-scrolling-subset-control: *truth-value*
no-stacked-data: *truth-value*
no-surface-polygon-borders: *truth-value*
nosmarttableplacement: *truth-value*
null-data-gaps: *truth-value*
null-data-value: *float*
null-data-value-x: *float*
null-data-value-z: *float*
object-in-server: *truth-value*
object-type: *integer*
ohlc-min-width: *integer*
old-scaling-logic: *truth-value*
overlap-multi-axes: *sequence*(*integer*)
page-height: *integer*
page-width: *integer*
painting: *truth-value*
percent-or-value-menu: *integer*
plot-method-menu: *integer*

plotting-method: *integer*
plotting-method-ii: *integer*
png-is-interlaced: *truth-value*
png-is-transparent: *truth-value*
png-transparent-color: *color*
point-colors: *sequence*(*color*)
point-hatch: *sequence*(*integer*)
point-label-rows: *integer*
point-labels: *sequence*(*text*)
point-padding: *float*
point-padding-area: *float*
point-padding-bar: *float*
point-size: *integer*
point-types: *sequence*(*integer*)
points: *integer*
points-to-graph: *integer*
points-to-graph-init: *integer*
points-to-graph-version: *integer*
polar-line-threshold: *float*
polar-tick-threshold: *float*
polar30-deg-threshold: *float*
poly-data *struct*
poly-mode: *integer*
prepare-images: *truth-value*
print-style-control: *integer*
quick-style: *integer*
quick-style-menu: *integer*
random-points-to-graph: *sequence*(*integer*)
random-subsets-to-graph: *sequence*(*integer*)
reset-gdicache: *truth-value*
right-edge-spacing: *float*
right-margin: *text*
rotation-detail: *integer*
rotation-increment: *integer*
rotation-menu: *integer*
rotation-speed: *integer*
ry-axis-color: *color*
ry-axis-comparison-subsets: *integer*
ry-axis-label: *text*
ry-axis-line-limit: *integer*
ry-axis-long-ticks: *truth-value*
ry-axis-scale-control: *integer*
ry-axis-whole-numbers: *truth-value*
sb-code: *integer*
sb-pos: *integer*
scale-for-rydata: *integer*
scale-for-txdata: *integer*

scale-for-xdata: *integer*
scale-for-ydata: *integer*
scale-for-zdata: *integer*
scale-symbols: *text*
scrolling-factor: *integer*
scrolling-horz-zoom: *truth-value*
scrolling-range: *integer*
scrolling-scale-control: *truth-value*
scrolling-subsets: *integer*
scrolling-vert-zoom: *truth-value*
separator-menu: *truth-value*
shadedpolygon-borders: *truth-value*
shading-style: *integer*
shadow-color: *color*
show-all-table-annotations: *truth-value*
show-annotations: *truth-value*
show-annotations-menu: *integer*
show-bounding-box: *integer*
show-bounding-box-menu: *integer*
show-contour: *integer*
show-contour-legends: *truth-value*
show-graph-annotations: *truth-value*
show-horz-line-annotations: *truth-value*
show-legend: *truth-value*
show-legend-menu: *integer*
show-pie-labels: *integer*
show-pie-legend: *truth-value*
show-polar-grid: *integer*
show-ry-axis: *integer*
show-table-annotation: *truth-value*
show-table-annotations-menu: *integer*
show-tick-mark-ry: *integer*
show-tick-mark-tx: *integer*
show-tick-mark-x: *integer*
show-tick-mark-y: *integer*
show-tx-axis: *integer*
show-vert-line-annotations: *truth-value*
show-x-axis: *integer*
show-x-axis-annotations: *truth-value*
show-y-axis: *integer*
show-y-axis-annotations: *truth-value*
show-z-axis: *integer*
show-z-axis-line-annotations: *truth-value*
simple-line-legend: *truth-value*
simple-point-legend: *truth-value*
slice-hatching: *integer*
slice-start-location: *integer*

smart-grid: *truth-value*
smith-chart: *integer*
special-date-time-mode: *truth-value*
special-scaling-ry: *integer*
special-scaling-y: *integer*
specific-plot-mode: *integer*
specific-plot-mode-color: *truth-value*
speed-boost: *integer*
spring-daylight *struct*
start-time: *float*
stop: *truth-value*
subset-by-point: *truth-value*
subset-colors: *sequence*(*color*)
subset-degree: *sequence*(*integer*)
subset-for-point-colors: *sequence*(*integer*)
subset-for-point-types: *sequence*(*integer*)
subset-hatch: *sequence*(*integer*)
subset-labels: *sequence*(*text*)
subset-line-types: *sequence*(*integer*)
subset-obstacles: *sequence*(*integer*)
subset-point-types: *sequence*(*integer*)
subset-shades: *sequence*(*color*)
subsets: *integer*
subsets-to-legend: *sequence*(*integer*)
subtitle: *text*
subtitle-bold: *truth-value*
subtitle-font: *text*
subtitle-italic: *truth-value*
subtitle-underline: *truth-value*
surface-polygon-borders: *truth-value*
symbol-frequency: *integer*
ta-border: *integer*
ta-grid-line-control: *integer*
taaxis-location: *integer*
taback-color: *color*
table-back-color: *color*
table-bmp-filename: *text*
table-bmp-style: *integer*
table-comparison-subsets: *truth-value*
table-font: *text*
table-fore-color: *color*
table-gradient-end: *integer*
table-gradient-start: *integer*
table-gradient-style: *integer*
table-what: *integer*
table-what-menu: *integer*
tacolor: *sequence*(*color*)

tacolumn-width: *sequence*(*integer*)
tacolumns: *integer*
tafont: *text*
tafonts: *sequence*(*text*)
tafore-color: *color*
taheader-column: *truth-value*
taheader-orientation: *integer*
taheader-rows: *integer*
tahot-spot: *sequence*(*integer*)
tajustification: *sequence*(*integer*)
talocation: *integer*
target-points-to-table: *integer*
tarows: *integer*
tatext: *sequence*(*text*)
tatext-size: *integer*
tatype: *sequence*(*integer*)
text-color: *color*
text-shadows: *integer*
tick-color: *color*
tick-style: *integer*
time-label-type: *integer*
top-margin: *text*
treat-comparisons-menu: *integer*
treat-comps-as-normal: *truth-value*
triangle-annotation-adj: *truth-value*
tx-axis-color: *color*
tx-axis-comparison-subsets: *integer*
tx-axis-label: *text*
tx-axis-line-limit: *integer*
tx-axis-long-ticks: *truth-value*
tx-axis-scale-control: *integer*
tx-axis-whole-numbers: *truth-value*
undo-zoom-menu: *integer*
upper-bound-text: *text*
upper-bound-value: *float*
vboundary-types: *integer*
vert-line-annot-hot-spot: *sequence*(*integer*)
vert-line-annotation: *sequence*(*float*)
vert-line-annotation-color: *sequence*(*color*)
vert-line-annotation-text: *sequence*(*text*)
vert-line-annotation-type: *sequence*(*integer*)
vert-orient90-degrees: *truth-value*
vert-scroll-pos: *integer*
vgnaxis-label-location: *truth-value*
viewing-height: *integer*
viewing-style: *integer*
viewing-style-menu: *integer*

wdesk-color: *color*
wgraph-back-color: *color*
wgraph-fore-color: *color*
working-axes-proportions: *sequence*(*float*)
working-axis: *integer*
working-table: *integer*
wshadow-color: *color*
wtable-back-color: *color*
wtable-fore-color: *color*
wtext-color: *color*
x-axis-annotation: *sequence*(*float*)
x-axis-annotation-color: *sequence*(*color*)
x-axis-annotation-text: *sequence*(*text*)
x-axis-color: *color*
x-axis-label: *text*
x-axis-line-limit: *integer*
x-axis-long-ticks: *truth-value*
x-axis-number-spacing: *float*
x-axis-on-top: *truth-value*
x-axis-scale-control: *integer*
x-axis-tick-spacing: *float*
x-axis-vert-numbering: *truth-value*
x-axis-whole-numbers: *truth-value*
xdata: *sequence*(*float*)
xzback-color: *color*
y-axis-annotation: *sequence*(*float*)
y-axis-annotation-axis: *sequence*(*integer*)
y-axis-annotation-color: *sequence*(*color*)
y-axis-annotation-text: *sequence*(*text*)
y-axis-color: *color*
y-axis-label: *text*
y-axis-line-limit: *integer*
y-axis-long-ticks: *truth-value*
y-axis-on-right: *truth-value*
y-axis-scale-control: *integer*
y-axis-vert-grid-numbers: *truth-value*
y-axis-whole-numbers: *truth-value*
yback-color: *color*
ydata: *sequence*(*float*)
year-label-type: *integer*
year-month-day-prompt: *integer*
z-axis-color: *color*
z-axis-label: *text*
z-axis-line-annotation: *sequence*(*float*)
z-axis-line-annotation-color: *sequence*(*color*)
z-axis-line-annotation-text: *sequence*(*text*)
z-axis-line-annotation-type: *sequence*(*integer*)

z-axis-long-ticks: *truth-value*
z-axis-whole-numbers: *truth-value*
zdata: *sequence*(*float*)
zdistance: *float*
zero-degree-offset: *integer*
zoom-interface-only: *integer*
zoom-max-axis: *integer*
zoom-max-ry: *float*
zoom-max-tx: *float*
zoom-max-x: *float*
zoom-max-y: *float*
zoom-min-axis: *integer*
zoom-min-ry: *float*
zoom-min-tx: *float*
zoom-min-x: *float*
zoom-min-y: *float*
zoom-mode: *truth-value*
zoom-style: *integer*

Known Chart View Enumeration Values

first-default-tab
peac-auto
peac-log
peac-normal
peadl-datapointlabels
peadl-datavalues
peadl-none
peadl-pointlabels
peae-allsubsets
peae-indsubsets
peae-none
peas-avgap
peas-avgpp
peas-m1sdap
peas-m1sdpp
peas-m2sdap
peas-m2sdpp
peas-m3sdap
peas-m3sdpp
peas-maxap
peas-maxpp
peas-minap
peas-minpp
peas-p1sdap
peas-p1sdpp
peas-p2sdap
peas-p2sdpp

peas-p3sdap
peas-p3sdpp
peas-pareto-asc
peas-pareto-dec
peas-sumpp
peaui-all
peaui-disablekeyboard
peaui-disablemouse
peaui-none
peaxd-include-sat-sun
peaxd-no-weekends
peaz-horizontal
peaz-horzandvert
peaz-none
peaz-vertical
pebfd-2nd
pebfd-3rd
pebfd-4th
pebg-transparent
pebs-bitblt-bottom-center
pebs-bitblt-bottom-left
pebs-bitblt-bottom-right
pebs-bitblt-center
pebs-bitblt-top-center
pebs-bitblt-top-left
pebs-bitblt-top-right
pebs-large
pebs-medium
pebs-no-bmp
pebs-small
pebs-stretchblt
pebs-tiled-bitblt
pecg-coarse
pecg-fine
pecg-medium
pecm-datacross
pecm-datasquare
pecm-floatingxonly
pecm-floatingxy
pecm-floatingy
pecm-floatingyonly
pecm-grayed
pecm-hide
pecm-nocursor
pecm-point
pecm-show
pecml-above-separator
pecml-below-separator
pecml-bottom
pecml-top
pecms-checked

pecms-unchecked
pecontrol-3d
pecontrol-graph
pecontrol-pgraph
pecontrol-pie
pecontrol-sgraph
pecpl-top-left
pecpl-top-right
pecps-none
pecps-xvalue
pecps-xyvalues
pecps-yvalue
pedlt-1-char
pedlt-3-char
pedlt-no-day-number
pedlt-no-day-prompt
pedlt-percentage
pedlt-value
pedo-driverdefault
pedo-landscape
pedo-portrait
pedp-disabled
pedp-enabled
pedp-inside-top
peds-3d
peds-none
peds-shadows
pedtm-delphi
pedtm-none
pedtm-vb
peedd-clipboard
peedd-file
peedd-printer
peesd-inches
peesd-millimeters
peesd-no-size-or-pixel
peesd-points
peetd-bmp
peetd-jpeg
peetd-metafile
peetd-png
peetd-text
pefs-large
pefs-medium
pefs-small
pefvp-auto
pefvp-horz
pefvp-slanted
pefvp-vert
pegam-not-moveable
pegam-pointer

pegat-addpolypoint
pegat-addtext
pegat-arrow-e
pegat-arrow-n
pegat-arrow-ne
pegat-arrow-nw
pegat-arrow-s
pegat-arrow-se
pegat-arrow-sw
pegat-arrow-w
pegat-bottomright
pegat-cross
pegat-dash
pegat-dashdotdotline
pegat-dashdotline
pegat-dashline
pegat-diamond
pegat-diamondsolid
pegat-dot
pegat-dotline
pegat-dotsolid
pegat-downtriangle
pegat-downtrianglesolid
pegat-ellipse-dash
pegat-ellipse-dashdot
pegat-ellipse-dashdotdot
pegat-ellipse-dot
pegat-ellipse-fill
pegat-ellipse-medium
pegat-ellipse-thick
pegat-ellipse-thin
pegat-endpolygon
pegat-endpolyline-dash
pegat-endpolyline-dashdot
pegat-endpolyline-dashdotdot
pegat-endpolyline-dot
pegat-endpolyline-medium
pegat-endpolyline-thick
pegat-endpolyline-thin
pegat-extraextrathinsolid
pegat-extrathicksolid
pegat-extrathinsolid
pegat-large-obstacle
pegat-largecross
pegat-largediamond
pegat-largediamondsolid
pegat-largedot
pegat-largedotsolid
pegat-largedowntriangle
pegat-largedowntrianglesolid
pegat-largeplus

pegat-largesquare
pegat-largesquaresolid
pegat-largeuptriangle
pegat-largeuptrianglesolid
pegat-linecontinue
pegat-medium-obstacle
pegat-mediumsolidline
pegat-mediumthicksolid
pegat-mediumthinsolid
pegat-nosymbol
pegat-nosymbol-movable
pegat-paragraph
pegat-pixel
pegat-plus
pegat-pointer
pegat-rect-dash
pegat-rect-dashdot
pegat-rect-dashdotdot
pegat-rect-dot
pegat-rect-fill
pegat-rect-medium
pegat-rect-obstacle
pegat-rect-thick
pegat-rect-thin
pegat-roundrect-dash
pegat-roundrect-dashdot
pegat-roundrect-dashdotdot
pegat-roundrect-dot
pegat-roundrect-fill
pegat-roundrect-medium
pegat-roundrect-thick
pegat-roundrect-thin
pegat-small-obstacle
pegat-smallcross
pegat-smalldiamond
pegat-smalldiamondsolid
pegat-smalldot
pegat-smalldotsolid
pegat-smalldowntriangle
pegat-smalldowntrianglesolid
pegat-smallplus
pegat-smallsquare
pegat-smallsquaresolid
pegat-smalluptriangle
pegat-smalluptrianglesolid
pegat-square
pegat-squaresolid
pegat-startpoly
pegat-starttext
pegat-thicksolidline
pegat-thinsolidline

pegat-topleft
pegat-uptriangle
pegat-uptrianglesolid
peglc-both
peglc-none
peglc-xaxis
peglc-yaxis
pegpm-area
pegpm-areastacked
pegpm-areastackedpercent
pegpm-bar
pegpm-barstacked
pegpm-barstackedpercent
pegpm-boxplot
pegpm-bubble
pegpm-contourcolors
pegpm-contourlines
pegpm-highlowbar
pegpm-highlowclose
pegpm-highlowline
pegpm-histogram
pegpm-horizontalbar
pegpm-horzbarstacked
pegpm-horzbarstackedpercent
pegpm-line
pegpm-openhighlowclose
pegpm-point
pegpm-pointsplusbfc
pegpm-pointsplusbfcgraphed
pegpm-pointsplusbfl
pegpm-pointsplusbflgraphed
pegpm-pointsplusline
pegpm-pointsplusspline
pegpm-ribbon
pegpm-specificplotmode
pegpm-spline
pegpm-step
pegpm-stick
pegpt-both
pegpt-graph
pegpt-table
pegs-dash
pegs-dot
pegs-horizontal
pegs-no-gradient
pegs-onepixel
pegs-thick
pegs-thin
pegs-vertical
pehs-annotation
pehs-bdiagonal

pehs-cross
pehs-datapoint
pehs-diagcross
pehs-fdiagonal
pehs-graph
pehs-horizontal
pehs-horzlineannotation
pehs-maintitle
pehs-multibottomtitle
pehs-multisubtitle
pehs-none
pehs-point
pehs-ryaxisgridnumber
pehs-subset
pehs-subtitle
pehs-table
pehs-tableannotation
pehs-tableannotation19
pehs-txaxisgridnumber
pehs-vertical
pehs-vertlineannotation
pehs-xaxis
pehs-xaxisannotation
pehs-xaxisgridnumber
pehs-xaxislabel
pehs-yaxis
pehs-yaxisannotation
pehs-yaxisgridnumber
pehs-yaxislabel
pehs-zaxisgridnumber
pehss-large
pehss-medium
pehss-scrolling-pointlabels
pehss-small
pehss-stationary-pointlabels
pelat-gridline
pelat-gridlineii
pelat-gridtick
pelat-gridtickii
pell-bottom
pell-left
pell-right
pell-top
pels-1-line
pels-1-line-inside-axis
pels-1-line-inside-overlap
pels-1-line-top-of-axis
pels-2-line
pelt-dash
pelt-dashdot
pelt-dashdotdot

pelt-dot
pelt-extraextrathinsolid
pelt-extrathicksolid
pelt-extrathinsolid
pelt-mediumsolid
pelt-mediumthicksolid
pelt-mediumthinsolid
pelt-thicksolid
pelt-thinsolid
pemas-group-all-axes
pemas-medium
pemas-none
pemas-separate-axes
pemas-thick
pemas-thickplustick
pemas-thin
pemc-grayed
pemc-hide
pemc-show
pemlt-1-char
pemlt-3-char
pemlt-no-month-prompt
pemps-large
pemps-medium
pemps-none
pemps-small
pemsc-max
pemsc-min
pemsc-minmax
pemsc-none
pemwf-horz-scroll
pemwf-no-scroll
pemwf-vert-scroll
peplm-lines
peplm-points
peplm-points-and-lines
peplm-surface
peplm-surface-w-contour
peplm-surface-w-pixels
peplm-surface-w-shading
peplm-wireframe
pepm-3dbar
pepm-polygondata
pepm-scatter
pepm-surfacepolygons
peps-large
peps-medium
peps-micro
peps-small
pepsc-current-style
pepsc-default-mono

pepsc-none
pept-arrow-e
pept-arrow-n
pept-arrow-ne
pept-arrow-nw
pept-arrow-s
pept-arrow-se
pept-arrow-sw
pept-arrow-w
pept-cross
pept-dash
pept-diamond
pept-diamondsolid
pept-dot
pept-dotsolid
pept-downtriangle
pept-downtrianglesolid
pept-pixel
pept-plus
pept-square
pept-squaresolid
pept-uptriangle
pept-uptrianglesolid
peptgi-firstpoints
peptgi-lastpoints
peptgv-random
peptgv-sequential
peqs-dark-inset
peqs-dark-line
peqs-dark-no-border
peqs-dark-shadow
peqs-light-inset
peqs-light-line
peqs-light-no-border
peqs-light-shadow
peqs-medium-inset
peqs-medium-line
peqs-medium-no-border
peqs-medium-shadow
peqs-no-style
perd-fulldetail
perd-plottingmethod
perd-wireframe
peri-decby1
peri-decby10
peri-decby15
peri-decby2
peri-decby5
peri-incby1
peri-incby10
peri-incby15

peri-incby2
peri-incby5
pesa-all
pesa-axislabels
pesa-empty
pesa-gridnumbers
pesa-labelonly
pesa-none
pesb-mouse-wheel-down
pesb-mouse-wheel-up
pesbb-always
pesbb-never
pesbb-whilerotating
pesc-admittance
pesc-bottomcolors
pesc-bottomlines
pesc-none
pesc-polar
pesc-rose
pesc-smith
pesc-topcolors
pesc-toplines
pesh-both
pesh-monochrome
pespl-label
pespl-none
pespl-percent
pespl-percentpluslabel
pespm-boxplot
pespm-highlowbar
pespm-highlowclose
pespm-highlowline
pespm-none
pespm-openhighlowclose
pess-colorshading
pess-financial
pess-none
pess-whiteshading
pesta-center
pesta-left
pesta-right
pestm-ticks-hide
pestm-ticks-inside
pestm-ticks-outside
petaal-bottom-center
petaal-bottom-full-width
petaal-bottom-left
petaal-bottom-right
petaal-bottom-table-spaced
petaal-new-row
petaal-top-center

petaal-top-full-width
petaal-top-left
petaal-top-right
petaal-top-table-spaced
petab-drop-shadow
petab-inset
petab-no-border
petab-single-line
petaho-270
petaho-90
petaho-horz
petaj-center
petaj-left
petaj-right
petal-bottom-center
petal-bottom-left
petal-bottom-right
petal-inside-axis
petal-inside-axis-0
petal-inside-axis-1
petal-inside-axis-2
petal-inside-axis-3
petal-inside-axis-4
petal-inside-axis-5
petal-inside-bottom-center
petal-inside-bottom-left
petal-inside-bottom-right
petal-inside-left-center
petal-inside-right-center
petal-inside-table
petal-inside-top-center
petal-inside-top-left
petal-inside-top-right
petal-left-center
petal-outside-axis
petal-outside-axis-0
petal-outside-axis-1
petal-outside-axis-2
petal-outside-axis-3
petal-outside-axis-4
petal-outside-axis-5
petal-overlap-axis
petal-overlap-axis-0
petal-overlap-axis-1
petal-overlap-axis-2
petal-overlap-axis-3
petal-overlap-axis-4
petal-overlap-axis-5
petal-right-center
petal-top-center
petal-top-left

petal-top-right
petlt-12hr-am-pm
petlt-12hr-no-am-pm
petlt-24hr
pets-1unit
pets-all-text
pets-bold-text
pets-dash
pets-dot
pets-gridstyle
pets-no-text
pets-thick
pets-thin
petw-allsubsets
petw-graphed
pevb-bottom
pevb-none
pevb-top
pevb-topandbottom
pevs-color
pevs-mono
pevs-monowithsymbols
pezio-line
pezio-normal
pezio-rect
pezs-framed-rect
pezs-ro2-not
solid-surface-color
wire-frame-color

Glossary

A

action: A command that G2 can execute within a rule, procedure, action button, or user menu choice.

annotation: The textual representation of compound attributes. For trend charts, an annotation is a detailed syntactical description of the non-default values of component attributes and component defaults.

argument: A value or item that is passed to an invoked procedure or function.

B

background: In G2's window, the visible pattern upon which G2 displays the current KB's workspaces. (Clicking the mouse on this background displays the Main Menu.) In a workspace, the visible color or image upon which the workspace's items appear to reside. (Clicking the mouse on this background displays the KB Workspace menu.)

buttons: Items that are among the components of a G2 application's user interface. Buttons perform actions or provide values for variables and parameters.

C

color pattern: A symbol list containing the name of each color region of an item, such as the background color of a workspace. Each region name is followed by the name of the color or metacolor of that region.

command line option: Included in the operating system command that launches a new G2 process, this keyword affects how the new G2 starts and runs.

component: For a trend chart, a building block that customizes a charting feature or a representation of data.

compound attribute: One or more components in a trend chart.

configuration: A declaration that changes the default behavior of items.

current scale: For a workspace, the scale at which G2 displays that workspace, which is a factor of its default scale. By specifying a new current scale, you can interactively or programmatically change the displayed size of a workspace. *Contrast with* default scale.

D

default color pattern: The color pattern specified in the class or icon definition of the item.

dense array a quantity array that can include elements whose value is 0. *Contrast with* sparse array.

E

element: A member of a list or array. Elements can be numeric values, truth values, symbols, text strings, or items.

element index: A numeric value specified within square brackets, used as a positional reference to a list or array element.

expression: A phrase that G2 evaluates to produce a value or a reference to an item.

extent: The visible, rectangular portion of a workspace. Also, the visible, rectangular region within which G2 displays an item's representation.

F

free text: A type of text item called to label various items in your KB. Free text lets you label your KB informatively and attractively.

G

G2 data interface object: The common term for a g2-to-g2-data-interface object.

G2 meter: A class that inherits from quantitative-variable and the mixin g2-meter-data-service. You can use such a class to instantiate G2-meters, and use these to measure how G2 uses time and memory.

Gensym character set: The characters that are valid to specify in a symbol or text value in G2. G2 provides facilities for its Text Editor that allow you to enter any character in the Gensym character set. When inputting or outputting symbol and text values, G2 also observes rules for translating those values, using the character codes of standard character sets.

GSI variable: A user-defined G2 variable subclass with gsi-interface as one of its direct superior classes. A GSI variable must specify a valid GSI interface object.

I

ICP (Intelligent Communications Protocol): Gensym's proprietary communications protocol, which allows G2s, G2 Gateway, and Telewindows to share information and distribute control among one or more G2 processes. ICP is a layer built on top of the TCP/IP networking protocol, depending on which platforms you are using in your network.

icon: The graphic representation of objects and items of other system-classes with an iconic representation style. In G2, items of many system-defined classes appear as icons. Use the Icon Editor to define the icon for user-defined classes.

indexed attribute: A user-defined attribute whose value G2 retrieves, using an internally maintained hash table. G2 can perform efficient searches to locate an item of a user-defined class by a particular value of one of its indexed attributes.

item layer position: For an item upon a workspace, the relative position of an item on top of or beneath other items that are upon the same workspace.

item layering: Whether items upon the same workspace appear on top of or beneath each other.

item passing: For G2 to G2, or G2 to a bridge process, the ability to pass a copy of any item as a reference, using a network handle.

K

keystroke: Pressing a key on your computer's keyboard, or pressing at the same time more than one key on your computer's keyboard that includes any combination of the Alt, Control, and Shift keys.

keystroke command: A keystroke that initiates a system-defined or user-defined operation in G2. G2 has many system-defined keystroke commands. With some constraints, you can use configurations to assign a system-defined or user-defined keystroke to either a system-defined or user-defined operation.

L

local name: A non-reserved symbol that represents an item or value in an expression. In rules and procedures, you can use implicit, or undeclared, local names. In procedures, you must declare local names that can receive an assignment. Use a local name to represent an item of any class or a value of any G2 type.

local window: The visible window that is a client of a G2 process. Among computers running the X Windows window manager, a G2 process's local window can be displayed either on the screen of the computer where G2 is running or on another computer's screen. *Contrast with* remote window.

M

matrix: an item-array whose elements are quantity-arrays. Each quantity-array represents one row of the matrix.

module: An entity in a knowledge base that G2 automatically associates with a set of system tables and optionally with a set of top-level workspaces. In this set of system tables, the Module Information system table names the module. If the current KB contains modules, it is *modularized*.

N

network handle: For item passing, an integer value, obtained by registering any item as a network entity.

O

object passing: The ability to pass a copy of an object from one G2 process to another via an external interface. Object passing is accomplished through the use of a remote procedure declaration to specify which attributes of the object to send.

P

permanent color pattern: The color-pattern of an item that is saved with the KB.

procedure: A list of statements that G2 can execute, either in sequence or concurrently, on zero or more arguments supplied when the procedure is invoked.

profile data: Information about the duration and order in which G2 performed an identified set of executable items in the current KB.

R

region: A named group of one or more icon layers. All the layers in a region have the same color. You can use the change action to change the color of any named region.

remote window: The visible window that is a client of a Telewindows process connected to a running G2 process. *Contrast with* local window.

representation: The appearance of an item, which is of a particular representation style.

required module: A module that contains items required by another module.

S

sparse array: An alternate way of expressing a dense array. A sparse array consists of two separate arrays:

- A quantity array, which holds only the non-zero elements of the corresponding dense array. This array is called the **value array**.
- An integer-array, which holds the index in the dense array of each non-zero element. This array is called the **index array**.

statement: An executable instruction in the body of a procedure.

subworkspace: A workspace that is subordinate to an item. Only items of certain classes can have a subworkspace, and an item of the proper class can have only one subworkspace. A subworkspace can be affected by the activatable-subworkspace configuration property on its superior item. (*See* activatable subworkspace.) *Contrast with* top-level workspace.

T

TCP/IP: A transport layer protocol for communications between computers. TCP/IP is one of two communication protocols supported by G2 and related Gensym products.

Telewindows: A Gensym product. When running, allows a user to connect to a running G2 process and view a window that displays the contents of the G2's current KB. Multiple Telewindows users can access one G2.

text-color color attribute: Determines the color of the text that appears in an item whose class has the text box representation style.

U

user mode: A non-reserved symbol that represents a category of usage or level of access to the current KB's knowledge. Declare a user mode simply by naming it in at least one configuration statement in at least one item in your KB. A G2 authorization file associates each authorized user name with a user mode.

W

warmboot: Resumption of execution of a KB after loading it from a snapshot file.

window: A display on a computer screen that is a client of a process, such as G2 or Telewindows. A window's user interface (for example, whether and how it can be moved, resized, or iconized) is determined by the window manager software in use on your computer.

workspace origin: The location with coordinates (0, 0) in workspace units. The origin of a new workspace is its center. After placing items upon a workspace, moving them, and shrink wrapping the workspace, the workspace's origin might no longer be within the workspace's displayed portion, or *extent*.

workspace unit: A unit of measure within a workspace that is equivalent to one pixel (one screen dot) when the workspace is shown at full size, which is proportionately larger or smaller as the workspace's scale is increased or decreased, respectively.

Index

Symbols

Numerics

A

B

C

D

E

F

G

g2-diff-texts
g2-directory-exists
g2-disable-change-logging-on-item
g2-disable-profiling
g2-disk-space-available-in-directory
g2-distribute-items
g2-drop-item-behind
g2-drop-item-to-bottom
g2-drop-workspace-behind
g2-drop-workspace-to-bottom
g2-enable-change-logging-on-item
g2-enable-profiling
g2-enter-package-preparation-mode
g2-enter-simulate-proprietary-mode
g2-execute-g2gl-process
g2-export-g2gl-process-as-xml
g2-export-g2gl-process-as-xml-text
g2-file-base-name-string
g2-file-device-string
g2-file-directory-list-to-string
g2-file-directory-string
g2-file-directory-string-to-list
g2-file-exists
g2-file-extension-string
g2-file-host-string
g2-file-names-are-identical
g2-files-in-directory
g2-file-version-string
g2-floating-client
g2-float-to-bytes
g2-float-to-text
g2-flush-change-log-for-entire-kb
g2-flush-change-log-for-item
g2-flush-version-information
g2-generate-new-random-seed
g2-get-all-g2gl-process-instances
g2-get-all-reserved-system-attribute-names
g2-get-and-log-performance-counter
g2-get-attribute-descriptions-of-class
g2-get-attributes-visible-in-mode
g2-get-attribute-texts-of-class
g2-get-attribute-values-of-item
g2-get-backward-chaining-rules-for-variable
g2-get-change-log-entry
g2-get-change-log-entry-comment
g2-get-class-hierarchy
g2-get-command-line-argument-from-launch
g2-get-connection-vertices
g2-get-containment-hierarchy
g2-get-default-error-handler
g2-get-default-item-color
g2-get-default-item-color-pattern
g2-get-description-of-item
g2-get-direct-subclasses
g2-get-environment-variable
g2-get-explanation-hierarchy
g2-get-floating-licenses-remaining
g2-get-font-of-text-box
g2-get-forward-chaining-focus-info
g2-get-g2-process-id
g2-get-g2-version-information
g2-get-g2-version-of-kb-file
g2-get-hash-table-value
g2-get-highest-from-priority-queue
g2-get-host-name
g2-get-item-color
g2-get-item-color-pattern
g2-get-item-color-regions
g2-get-item-from-network-handle
g2-get-item-layer-position
g2-get-items-connected-to-port
g2-get-items-in-area
g2-get-item-with-uuid
g2-get-logbook-message-handler
g2-get-matrix-dimensions
g2-get-maximum-height-of-text-box
g2-get-maximum-width-of-text-box
g2-get-message-board-message-handler
g2-get-method-hierarchy
g2-get-modes-for-authorized-user
g2-get-module-hierarchy
g2-get-movement-limits
g2-get-network-address-list
g2-get-network-handle-from-item
g2-get-network-type
g2-get-network-type-given-index
g2-get-performance-counter
g2-get-performance-frequency
g2-get-permanent-item-color
g2-get-permanent-item-color-pattern
g2-get-port-number-or-name
g2-get-port-number-or-name-given-index
g2-get-position-of-element-in-array
g2-get-position-of-element-in-list
g2-get-position-of-element-in-sequence
g2-get-priority-in-queue
g2-get-procedure-caller-hierarchy
g2-get-procedure-calling-hierarchy
g2-get-procedure-invocation-hierarchy
g2-get-profiled-information
g2-get-random-seed
g2-get-reflection-and-rotation

g2-get-software-version
g2-get-strict-instances-of-class
g2-get-subscription-handle-info
g2-get-subscription-handles-for-items
g2-get-sys-mod-version
g2-get-text-extent
g2-get-text-of-trend-chart-component
g2-get-top-level-workspaces
g2-get-tracked-item-call-sequence
g2-get-tracked-items-call-depth
g2-get-window-license-type
g2-get-workspace-hierarchy
g2-get-workspace-layer-position
g2-hash-table-to-sequence
g2-import-g2gl-process-from-xml
g2-import-g2gl-process-from-xml-text
g2-indexed-attribute-item-list
g2-initialize-parameter
g2-insert-in-priority-queue
g2-invoke-web-service-operation
g2-is-in-package-preparation-mode
g2-is-in-simulate-proprietary-mode
g2-item-has-do-not-strip-text-mark
g2-item-has-strip-text-mark
g2-item-is-editable
g2-item-is-showing-in-window
g2-kill-all-g2gl-process-instances
g2-kill-g2gl-process-instance
g2-kill-process
g2-kill-remote-process
g2-last-input-event
g2-last-input-event-info
g2-latest-date-file-attributes-were-changed
g2-latest-date-file-was-accessed
g2-latest-date-file-was-modified
g2-leave-package-preparation-mode
g2-leave-simulate-proprietary-mode
g2-length-of-file
g2-lift-item-in-front-of
g2-lift-item-to-top
g2-lift-workspace-in-front-of
g2-lift-workspace-to-top
g2-load-kb
g2-lu-back-substitute
g2-lu-decompose
g2-lu-solve
g2-make-uuid
g2-make-workspaces-proprietary-now
g2-matrix-multiply
g2-measure-memory
g2-merge-kb
g2-merge-kb-ex
g2-move-from-area-of-workspace
g2-move-items-to-mouse
g2-move-objects
g2-name-for-item
g2-nms-add-break
g2-nms-add-choice
g2-nms-add-control
g2-nms-add-right-justifier
g2-nms-add-separator
g2-nms-add-submenu
g2-nms-check
g2-nms-create-combo-box
g2-nms-create-edit-box
g2-nms-create-menu
g2-nms-create-menu-bar
g2-nms-create-submenu
g2-nms-create-toolbar
g2-nms-delete-all-menu-choices
g2-nms-delete-control
g2-nms-delete-menu
g2-nms-disable
g2-nms-dismiss
g2-nms-enable
g2-nms-get-built-in-menu
g2-nms-get-callback
g2-nms-get-choices
g2-nms-get-choice-with-key
g2-nms-get-colors
g2-nms-get-help
g2-nms-get-icon
g2-nms-get-key
g2-nms-get-label
g2-nms-get-menu-bar
g2-nms-get-menus
g2-nms-get-posting-callback
g2-nms-get-selection-callback
g2-nms-hide-menu-bar
g2-nms-is-checked
g2-nms-is-choice
g2-nms-is-enabled
g2-nms-is-menu
g2-nms-is-menu-bar
g2-nms-is-supported
g2-nms-manage-popup-menu
g2-nms-radio-check
g2-nms-reset
g2-nms-set-callback
g2-nms-set-choice
g2-nms-set-colors
g2-nms-set-help

H

I

K

L

M

Q

R

S

T

U

V

W

X