G2-PI Bridge

User's Guide

Version 2020

G2-PI Bridge User's Guide, Version 2020

May 2020

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

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

Copyright © 1985-2020 Gensym Corporation

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

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

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

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

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

Copyright © 1998-2002 Daniel Veillard. All Rights Reserved.

SCOR® is a registered trademark of PRTM.

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

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

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

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

Part Number: DOC093-1200

Contents

Preface

Describes this guide and the conventions that it uses.

About this Guide

This guide describes the G2-PI Bridge, which allows you to access PI data from a G2 application.

Version Information

The G2-PI Bridge is only available on Windows platforms.

Audience

This guide is for developers who must retrieve information from PI for use within G2 applications and optionally must write information from G2 into PI. It assumes at least a limited understanding of PI and a basic knowledge of G2.

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

Overview of the G2-PI Bridge

Describes the architecture of the G2-PI Bridge.

Introduction

The G2-PI Bridge allows your G2 application to access the data points monitored by a PI™ (Plant Information) system. In installations already using PI, the bridge is a very convenient and cost-effective way of making this data available to your G2 system.

PI performs plant-wide monitoring and analysis by handling the collection and storage of numerical data from various sensors in an industrial process. The data can be used locally within the PI system, for display and logging, as well as transmitted to G2 via the G2-PI Bridge.

The G2-PI Bridge provides the following features:

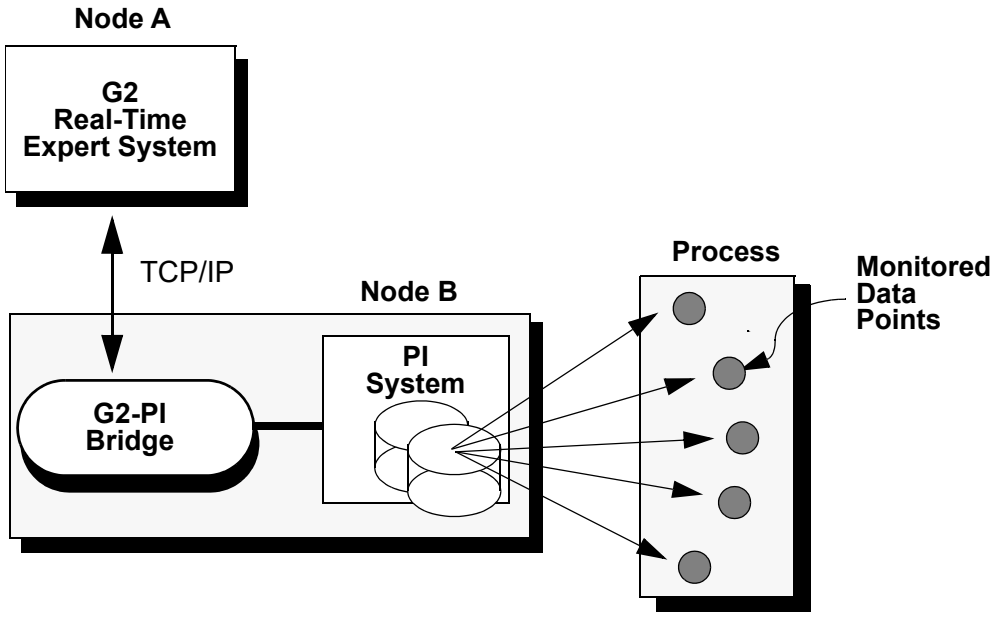
- Two-way data transfer between G2 applications and PI systems.
- Communication with multiple PI systems through the same bridge process.
- Support for many G2 data types, facilitated by flexible data conversions.
- Extensive error logging and troubleshooting capabilities.

The G2-PI Bridge allows two ways of accessing PI data:

- Using G2 variables. These variables can be of the classes osipi-int, osipi-real, or osipi-digital, which are defined in the `g2-pi.kb` knowledge base. You can also define your own classes for storing PI data.
- Using remote procedure calls (RPCs) to read the attributes of PI tags and to obtain historical data stored in PI. You can use this data for trend analysis and for sophisticated inferencing in G2.

The G2-PI bridge uses PI-API to exchange information with PI via TCP/IP. PI-API must be installed on your system to use the G2-PI bridge. PI-API is a product from OSIsoft.

A typical configuration might look like this:



2

Starting the G2-PI Bridge

Describes how to run the G2-PI Bridge under default conditions.

Introduction

Before you can start the G2-PI Bridge, you must authorize the bridge. If you are accessing data from a single PI server, you can simply start the bridge from the command-line. If you are accessing data from multiple PI servers, you must first configure the data file that tells the bridge about the available servers, and optionally which user names and passwords to use.

You start the G2-PI Bridge from the command line. You can also specify the port, a configuration file, a log configuration file, and command-line options.

Authorizing the G2-PI Bridge

Before you can run the G2-PI Bridge, you must get authorization codes for the G2 Bundle or the `gsi.ok` file. Contact Order Services for these codes at 1-781-265-7106.

Using a Single Server

If you are reading PI data from a single server, there is no need to configure the server data file. The G2-PI Bridge will use the default server. When using the default server, there is no need to use any special syntax to refer to PI data points or tags.

Configuring Multiple Servers

If you are reading PI data from multiple servers, or if you require password-protected access to the data, you must create a data file for the bridge that contains information about the servers, user names, and passwords. The default name of this file is `pisrvrs.dat`. Use the program `g2piconfig.exe` to create and edit this data file.

To access PI data from a particular server from within G2, you must precede the PI tag name with the server name and a colon.

Creating the Server Data File

By default, the bridge looks in the directory from which you started the bridge for the server data file with this name:

```
pisrvrs.dat
```

This file exists in the `pi` directory of your G2 installation directory.

To specify the exact name and location of the server data file, create an environment variable named `G2PSL` and specify the name and location there. For example:

```
set G2PSL=C:\Gensym\PI\servers.pbf
```

Configuring the Servers

If you have more than one server and you use the default user names and passwords for all of them, you can use either the program `g2piconfig.exe` or a normal text editor to configure the server data file. If you use `g2piconfig.exe`, leave the user name and password fields blank. If you use a text editor enter a list of servers with carriage returns between each server, for example:

```
PIServer1
PIServer2
PIServer3
```

Configuring Users and Passwords

If you are configuring servers, as well as users and passwords, each entry of the server data file should consist of a server name, a user name, and an encrypted password, separated by commas, for example:

```
PIServer1,user1,encyrpted-password
PIServer2,user2,encyrpted-password
PIServer3,user3,encyrpted-password
```

You can only associate one user name with a given server; this is a limitation of PI-API.

Because you cannot use a text editor to enter encrypted passwords, you must use the program named `g2piconfig.exe` to edit the server data file. This file is located in the `pi` directory of your G2 installation directory.

The program allows you to insert, edit, and delete server descriptions. You can use menu commands and popup menu commands to perform these functions. You can also use the Insert, Delete, and Enter keys to edit the server descriptions.

Starting the Bridge Process

You invoke the G2-PI Bridge from the command line, using the following syntax:

```
g2pi [port-number] [@configuration-file | $configuration-file]
[-option ...]
```

If you do not provide a *port-number*, the bridge will attempt to use the default port number 22041. If the default port is not available, the bridge will attempt to use the first available port within the next 99 addresses. If you provide a port number, it must be the first argument after `g2pi`.

The G2-PI Bridge can read configuration options from a configuration file or directly from the command line. You can use either the `@` symbol or the `$` symbol, followed by the configuration file name, to load any set of command-line options. For information on the format of this file, see Using a Configuration File.

You can also enter command-line options directly on the command line when starting the bridge, as described in Using Command-Line Options.

You can specify the configuration file and command-line options in any order. If you specify contradictory options such as “log to file” and “do not log to file,” the last option encountered by the initialization process takes precedence.

Using Command-Line Options

As Configuring Connections describes, connections between G2 and the G2-PI Bridge are controlled by a G2 object called an interface object. You may have up to

49 interface objects in a G2-PI application. Each connection defines what is called a context. You can think of a context as an environment that defines some behavior such as which errors are reported and how. Different contexts can behave differently.

Caution Attempting to make all 49 connections at the same time can cause the bridge to abort.

You can use command-line options and configuration files to configure connection and logging parameters for the bridge process at startup. In some cases, a command-line option specifies general bridge behavior. In others, it specifies the default behavior of context-specific options. Individual contexts can select non-default behavior by using either initialization-string options or RPCs. For details, see Remote-process-initialization-string and Logging Operations.

Command-Line Options

The following table describes command-line options that you can include on the command line when starting the bridge. Each option below that accept values of `y` or `n` (but not `y/n/a`) sets the default behavior of all contexts and can be overridden by individual contexts with initialization-string options and RPCs.

For command-line options with arguments, a space between the command-line option and the argument is optional. For example, the following pairs of command-line options are equivalent:

`-by` and `-b y`

`-d5` and `-d 5`

`-llog.log` and `-l log.log`

`-m2M` and `-m 2M`

Command-Line Option	Description
`-help`	Displays G2-PI Bridge command-line options and does not start the bridge process.
`-tcpipexact`	Specifies the exact TCP/IP port number to use when starting the bridge. If the bridge cannot listen on the specified port, it will not attempt to use another port and the process will halt.
`-b[y/n]`	Determines whether to log output to the G2 Message Board. Default is `n`.
`-d[0-9, y/n]`	Sets the debugging level. `0` means the fewest messages; `y` means a debug level of 9; `n` means a debug level of 2. Default is `2`.
`-f[y/n]`	Determines whether log messages sent to a file are buffered (`n`) or written immediately and flushed (`y`). Default is `n`.
`-g[y/n]`	Determines whether log messages are sent to the G2 procedure named osi-error. Default is `y`. For more information, see Logging Operations.
`-l` *log-file-name*	Sets the name of the log file. Default is `g2pi.log`.
`-m` *size*	Sets the maximum log file size. 0 means as large as possible. The *size* can use K for kilobytes (1,024) or M for megabytes (1,048,576). The minimum is `1K`. The default is `0`.
`-n[y/n/a]`	Determines what to do with an existing log file when starting a new logging session. `y` means overwrite the log file; `n` means rename the log file by appending a file number to the file name before creating a new log file; and `a` means append messages to the old log file. Default is `a`.
`-o[y/n]` `-s[y/n]`	Determines whether to log output to the screen. Default is `y`.
`-p[y/n]`	Determines whether to log output to the PI log file. Default is `n`.

Using a Configuration File

A configuration file is a text file with one command-line option on each line, up to a maximum of 25 commands. The command-line options use the same syntax as described in Using Command-Line Options.

You load a configuration file by using either the `@` or `$` command-line option.

For example, a configuration file might look like this:

```
-by
-d9
-fy
-gy
-l my-log.log
-m2M
-na
```

3

Configuring Connections

Describes how to configure attributes of the GSI interface object.

Introduction

To access PI data from within G2, you must create and configure a GSI interface object. The interface object specifies the connection configuration information, which includes the name and port of the computer on which G2-PI Bridge is running.

Every variable that uses the G2-PI Bridge to retrieve values from PI has an attribute that contains a reference to a PI point. This attribute is called the **PI pointer**. The interface object specifies the name of the PI pointer. Optionally, it specifies the attribute of the variable that contains the details about which PI attributes to retrieve and where to store them. The interface object can override default logging behavior. It specifies the timeout period and whether the external system updates the data or whether G2 polls the external system to update the data. To establish a connection between G2 and PI, you simply configure and enable the interface object.

The interface object reports its connection status in a read-only attribute.

Once a connection is established, you use the interface object to connect G2 variables to data sources in the bridge. The G2 application can also call remote procedures across the interface.

The GSI interface object serves as the connecting point to the bridge process. All communication with the bridge takes place through this object.

A Note on Terminology

A **variable** is a type of G2 object. PI has a similar object, known as a **PI point**.

In object-oriented terms, a data field of an object is called an **attribute**. Thus, Gensym calls the fields of interface objects and variables "attributes," and OSIsoft calls the fields of PI points "attributes." The G2-PI Bridge works with both types of attributes. When this manual refers to an "attribute," its type will be explicitly stated or it will be clear from the context.

A **PI variable** is a G2 variable that has been configured to store values retrieved from PI by the G2-PI Bridge.

Whereas "PI tag" and "PI point" are often used interchangeably, technically they are different. A **PI point** is a PI structure that holds a current value and numerous attributes. A **PI tag** is the name of a PI point.

Creating and Configuring GSI Interface Objects

To connect from G2 to the PI bridge, you can use a standard GSI interface object, or you can use an osipi_interface object, which is defined in the `g2-pi.kb` knowledge base. The advantage of using an osipi_interface object is that it has a unique icon, which changes its color to green when it successfully connects to the G2-PI Bridge. Otherwise, the two types of interface objects are identical.

In general, you should merge `g2-pi.kb` into your knowledge base, because it contains the remote procedure declarations for all the RPC functions that you can use. It also defines several classes that you can use to simplify configuration. The descriptions below assume that `g2-pi.kb` has been merged into your application. However, if you created your knowledge base with Version 4.0 of the bridge, if your application does not contain the g2-pi module, and if it does not need any of the Version 5.0 RPC function, then you do not need to merge `g2-pi.kb` into your knowledge base.

Creating the GSI Interface Object

To create a GSI Interface object:

➔ Choose KB Workspace > New Object > network-interface > gsi-interfaces > gsi-interface.

or

➔ Choose KB Workspace > New Object > network-interface > gsi-interfaces > osipi_interface.

If you have not included the g2-pi module in your application, you will not be able to create an osipi-interface. In this case, to create a GSI interface object, choose KB Workspace > New Object > network-interface > gsi-interface.

As you can see from the following tables, the two types of interface objects are identical, except for their icons and class names. Thus, the descriptions in the rest of this chapter apply equally to either type of object.

a gsi-interface	
Notes	OK
Names	none
Identifying attributes	none
Interface warning message level	default to warning message level
Disable interleaving of large messages	no
Interface timeout period	use default
Interface initialization timeout period	unlimited
GSI connection configuration	none
External system has a scheduler	no
Poll external system for data	no
Grouping specification	no grouping
Remote process initialization string	""
GSI application name	default
GSI interface status	0
Interval to poll external system	use default

an osipi_interface	
Notes	OK
Names	none
Identifying attributes	none
Interface warning message level	default to warning message level
Disable interleaving of large messages	no
Interface timeout period	use default
Interface initialization timeout period	unlimited
GSI connection configuration	none
External system has a scheduler	no
Poll external system for data	no
Grouping specification	no grouping
Remote process initialization string	""
GSI application name	default
GSI interface status	0
Interval to poll external system	use default

Configuring the GSI Interface Object

Now, you must configure the attributes of the GSI interface object as described below. You only need to configure the attributes described below.

Names

The name of the GSI interface object, which must be unique within G2. You must name the interface object. Because this object represents the connection to a particular G2-PI Bridge, choose a name that associates it with the PI system.

Gsi-connection-configuration

Describes the network connection between G2 and the G2-PI Bridge. It specifies the type of network and the network address of the bridge process. The G2-PI Bridge supports TCP/IP only.

Note Editing this attribute automatically establishes a connection between G2 and the bridge.

The syntax for specifying the location of the bridge process is:

tcp-ip host "*host-name*" port-number *port-number*

where:

"*host-name*"	The name of the machine running the bridge process. Note that the host name is enclosed in double quotation marks.
port-number	The port number of the process with which the bridge process is started. You set the port number from the command line when you start the bridge process. The default port is 22041. You can change it to any number over 3000 and under 30,000 that you are not using for another process on that machine.

For example:

tcp-ip host "localhost" port-number 22044

Identifying-attributes

Every variable that is to receive values from PI must contain an attribute that refers to the PI tag that is the source of the variable's value. For example, if you create a variable of type osipi-real and you want it to get its value from the SINUSOID tag in the PI server JUPITER, you must set the osipi-tagname attribute of the OSIPI variable to "JUPITER:SINUSOID". The variable has an attribute that

refers to a PI tag and, in this case, the name of this attribute is osipi-tagname. G2 removes the hyphen for display purposes.

This attribute is known as the PI pointer. If you define your own PI variable class, the PI pointer may have a different name. However, regardless of its name, it must exist. See Referring to PI Tag Names. The first and most important use of identifying-attributes in an interface object is to specify the name of the PI pointer.

Optionally, you can request that the PI bridge retrieve the values of PI attributes and store them in attributes of your variable. To do this, your variable must define an attribute whose value is a structure that specifies the PI attributes to retrieve and the attributes of the variable in which they should be stored.

Next, you must append to the identifying-attributes of the interface object a comma followed by the name of the attribute that contains the defining structure. For example, if you define an attribute named pi-attributes-structure, the value of the identifying-attributes would be:

```
osipi-tagname, pi-attributes-structure
```

If the identifying-attributes contains the name of a variable attribute that contains a structure, the PI bridge will try to retrieve PI attributes; otherwise, it will not.

For more information, see Accessing PI Data in G2.

External-system-has-a-scheduler

If a variable has a default-update-interval other than none and external-system-has-a-scheduler is set to no in the associated interface object, the bridge will request from PI a refresh of the value of the variable as soon as the default-update-interval expires. If external-system-has-a-scheduler is set to yes, the bridge will only request a refresh of the value if the variable's validity-interval has expired and G2 has requested the value via a rule, a readout table, or some other means.

Note that setting this value to yes may make it appear that a variable is not being updated. However when G2 requests the variable's value, the variable will be properly updated.

Poll-external-system-for-data

If the variables connected to this interface object will be updated by exception reporting, set this value to yes. Otherwise, set it to no. Setting this value to yes will cause the bridge to regularly poll PI to see if it has an exception report for it.

For more information, see Exception Reporting.

Grouping-specification

In the G2-PI Bridge, always set this attribute to no grouping. All PI points are independent, therefore, there is no benefit to grouping them.

Remote-process-initialization-string

Many of the command-line options, which you enter when you start the bridge process, set the default behavior. You can override these defaults by using similar initialization string options. For example, if you start the bridge with the command-line option `-on`, by default, the bridge does not show error messages on the screen. If you specify –oy in the remote-process-initialization-string of an interface object, errors associated with variables connected to that interface object will be displayed on screen.

You can specify the following options in the remote-process-initialization-string attribute of a particular GSI interface object to configure the behavior for an individual context. The options use a similar syntax to that of the command-line options as described in Using Command-Line Options.

The initialization string options are shown in the following table. Successive options should be separated by one or more spaces. A space between an option and its parameter is optional.

Command	Description
-b[y/n]	Determines whether to log output to the G2 Message Board for this context. Default is `n`.
-d[0-9, y/n]	Sets the debugging level for this context. `0` means the fewest messages; `y` means a debug level of 9; `n` means a debug level of 2. Default is `2`.
-e[y/n]	Sends a value to a PI point. The syntax of the set command references a variable, which references, in turn, a PI point. When this option is set (y), the variable will be set to the new value at the same time that the value is sent to PI. Otherwise (n), the value of the variable will not be changed until the next time a normal update occurs. Default is `n`.
-g[y/n]	Determines whether to log messages to the G2 procedure named osi-error for this context. Default is `n`.
-l[y/n]	Determines whether to send messages for this context to the log file.
-o[y/n]	Determines whether to log output to the screen for this context. Default is `y`.
-p[y/n]	Determines whether to log output for this context to the PI log. Default is `n`.

Command	Description
-t[y/n]	Determines whether timestamps will take their values from PI (y) or from the local computer (n), for this context. Default is n.
-x[y/n]	Determines whether to use exception reporting for this context. Default is n.

The following table shows which initialization string options you use to override which command-line options. The default value specifies the behavior if neither the command-line option nor the initialization string option is present.

This initialization string option...	Overrides this command-line option...	And the default is...
-b	-b	n
-d	none	2 (n)
-e	none	n
-g	-g	y
-l	none	n
-o	-o and -s	y
-p	-p	n
-t	none	n
-x	none	n

You can also use RPC functions to set or retrieve log configuration options for a particular context. For more information, see Logging Operations.

Interface-timeout-period

The length of time G2 will wait for a response from the bridge before reporting an error. A reasonable setting depends on how busy your network is and other factors. A good starting value is between 10 and 20 seconds. The minimum setting is 1 second.

Gsi-interface-status

An attribute that is set by G2 and indicates the status of the connection between G2 and the G2-PI Bridge. For details, see Determining the Connection Status.

Interval-to-poll-external-system

Determines how often G2 polls the bridge to determine if PI has a report of a changed value. This attribute is only used when poll-external-system-for-data is set to yes. The default is 1 second. You should not change this attribute.

Connecting to the Bridge Process

Once you have configured an interface object, you connect G2 to the G2-PI Bridge to begin receiving data from PI. G2 connects to the bridge automatically under these conditions:

- If the interface object is properly configured and enabled, starting G2 causes the connection to be made.
- If G2 is running and the interface object is properly configured but disabled, enabling the interface object causes the connection to be made.
- If G2 is running and the interface object is enabled, editing its gsi-connection-configuration attribute causes any existing connection to be broken, then causes a new connection to be made, using the new connection information.

Determining the Connection Status

G2 reports the connection status in the gsi-interface-status attribute of the interface object. The possible status values are:

2 (OK) The connection between the G2 process and the bridge process is successful and is being maintained.

1 (Initializing) The PI system is initializing. When G2 receives this code, it suspends sending messages to the bridge process until it receives an OK code.

0 (Waiting) The interface is either disabled or inactive.

−1 (Timeout) The G2 process has not heard from the bridge process within the interface-timeout-period specified for the interface object; thus, the connection has timed out. This code may also indicate that a communications overload has occurred. It is not necessarily an alarm condition because the bridge status usually returns to 2 without intervention.

−2 (Error) An error condition occurred. The connection between G2 and the G2-PI Bridge process has been interrupted.

4

Accessing PI Data in G2

Describes how to use GSI variables to access PI data in G2.

Introduction

The G2-PI Bridge can retrieve three types of PI data:

- Current values.
- PI point attributes.
- Historical values.

You store the retrieved values in G2 variables. The PI bridge retrieves current values and PI point attributes automatically after you configure the interface object and the variable. To retrieve historical values, you must use Remote Procedure Calls (RPCs).

The automatic update of a variable's value can occur in one of two ways:

- G2 can request the value.
- PI can inform G2 when a value changes more than a preset amount stored in PI point attributes. This technique is called **exception reporting**.

You can also use Remote Procedure Calls to read individual PI attributes. For details, see the description of get-property in Item Operations.

Retrieving PI Data

Retrieving the three PI data types has various configuration requirements, some of which are common and others of which depend on the type of PI data you are retrieving.

Common Requirements for Retrieving PI Data

The following configuration requirements apply when retrieving PI data of any of the three types:

- The gsi-interface-name attribute of every variable must contain the name of a properly configured interface object. The configuration of the interface object determines the behavior of the variable. For example, if the exception report option (-xy) is included in the remote-procedure-initialization-string of an interface object, all variables that use that interface object will be updated by exception. For details, see Creating and Configuring GSI Interface Objects.
- The identifying-attributes of the interface object must contain the name of the attribute of the variable that is the PI pointer. For details, see Identifying-attributes.

Retrieving Current Values

When retrieving current values, the PI pointer determines the PI point to read. The PI pointer is of the form "tagname" or "servername:tagname". For details, see Referring to PI Tag Names.

The configuration of the interface object determines whether or not the variable is updated by exception. The configuration of the PI variable depends on whether exception reporting is enabled.

For more information, see:

- Creating and Configuring GSI Interface Objects
- Using OSIPI Variables.
- Exception Reporting.

Non-Exception Configuration

If you do not want the variable to be updated by exception:

- Configure these attributes of the interface object as follows:

external-system-has-a-scheduler	In general, we recommend that you set this attribute to no to avoid update behavior that is different from what is expected.
poll-external-system-for-data	Set to no.
remote-process-initialization-string	Ensure that –xy is not included.

- Configure the variable as follows:

validity-interval	Set to something other than indefinite.
default-update-interval	Set to something other than none.

Exception Report Configuration

If you want the variable to be updated by exception:

- Configure the interface object as follows:

external-system-has-a-scheduler	This attribute it not used because you are going to set the default-update-interval of the variable to none. Set to yes.
poll-external-system-for-data	Set to yes.
remote-process-initialization-string	Include –xy.

- Configure the variable as follows:

validity-interval	Set to indefinite.
default-update-interval	Set to none.

Retrieving PI Attributes

To retrieve PI attributes, the identifying-attributes of the interface object must contain a second name after the name of the PI pointer. The second name is the name of an attribute of the variable that contains a structure, which specifies the PI attributes to retrieve and where they should be stored.

For more information, see PI Point Attributes.

Retrieving Historical Values

If you are using one of the predefined PI variable types in `g2-pi.kb`, all you need to do to retrieve historical data is to call get-time-vals or get-interp-vals.

If you have defined your own PI variable class, to retrieve historical data:

- You must ensure that you have defined attributes to hold the historical values.
- Depending upon what you named your history attributes, you might need to call rpc-define-history-attributes.
- You must call get-time-vals or get-interp-vals.

For more information, see Retrieving Historical Values

Using OSIPI Variables

You can use one of the predefined variable classes in the `g2-pi.kb` to store the values you retrieve from PI, or you can define your own PI variable classes. Note that if you are storing PI attributes in a variable, you must define your own variable class.

Using the OSIPI Variable Classes

The G2-PI Bridge KB provides these classes of variables:

- **osipi-var** is a subclass of the G2 quantitative-variable class. It has two subclasses:
  - **osipi-real** is used for floating point values.
  - **osipi-int** is used for integer values.
- **osipi-digital** is a subclass of the G2 text-variable class

In general, osipi-real and osipi-int variables behave just like other G2 float variables and integer variables. You can use them as objects or as data types for attributes of more complex objects. For example, you might create an osipi-real or an osipi-int object for each PI point in your application, then connect each of these objects to your flow diagram. You might also define more complex classes that

use osipi-real and osipi-int as the data types for attributes that get their values from the PI system.

The following example shows the table for a pump object on the left, which gets its data for the inflow attribute from a PI system. The inflow attribute is assigned the type osipi-real. The subtable for the inflow attribute on the right relates the inflow attribute to a particular PI system and PI point by specifying the gsi-interface-name and osipi-tagname.

a centrifugal-pump	
Notes	OK
Names	none
In service	1
Recent alarm time 1	****
Recent alarm time 2	****
In alarm	false
Max flow	60
Pump delta p	20
Min p to pump	15.7
Cv equivalent when off	0.0
Df dp in	1
Inflow	6.5
Outflow	****
P in	****
P out	****
Calc pump delta p	****
P in guess	****
P out guess	****
Liquid present flag	****
Inferred status	****

an osipi-real, the inflow of some centrifugal-pump	
Options	do not forward chain, breadth first backward chain
Notes	OK
Names	none
Tracing and breakpoints	default
Data type	quantity
Initial value	none
Last recorded value	6.5, expires 28 Apr 95 11:00:20 a.m.
History keeping spec	do not keep history
Validity interval	10 seconds
Formula	none
Simulation details	no simulation formula yet
Initial value for simulation	default
Data server	GSI data server
Default update interval	4 seconds
GSI interface name	osipi_one
GSI variable status	0
Object index	2
Pi point	1
Osipi tagname	"CU:002"
Osipi data type	"R"
Timed value	a pi-quantity-list
Timedate text	a pi-text-list
Timedate seconds	a pi-quantity-list
Last recorded pi time	"04281995100010"

Attributes of osipi-int and osipi-real

The osipi-int and osipi-real classes inherit attributes from G2 quantitative-variable and gsi-data-service, and they define additional attributes for interfacing with PI points.

Here is the table for an instance of osipi-int that uses the GSI interface object named osi_pi1 to get values from the PI point named 001 on the default server:

OSIPI-INT1, an osipi-int	
Options	do not forward chain, breadth first backward chain
Notes	OK
Names	OSIPI-INT1
Tracing and breakpoints	default
Data type	quantity
Initial value	none
Last recorded value	no value
History keeping spec	do not keep history
Validity interval	indefinite
Formula	none
Simulation details	no simulation formula yet
Initial value for simulation	default
Data server	GSI data server
Default update interval	none
GSI interface name	osi_pi1
GSI variable status	0
Data server for messages	gsi-data-server
Object index	0
Pi point	0.0
Osipi tagname	"001"
Osipi data type	"I"
Timed value	a pi-integer-list
Timedate text	a pi-text-list
Timedate seconds	a pi-quantity-list
Last recorded pi time	""

Attributes inherited from osipi-var

The following table describes the PI-specific attributes and the inherited attributes with special relevance. For information about the other attributes, see Chapter 15, "Variables and Parameters" in the *G2 Reference Manual*.

Attribute	Description
validity-interval	The length of time that the last recorded value should remain current. The setting depends on whether exception reporting is enabled. See Exception Reporting. When exception reporting is enabled, a new value will be automatically transmitted by the G2-PI Bridge when a change exceeds preset limits. Thus, you should set this attribute to indefinite. When exception reporting is disabled: • If external-system-has-a-scheduler is set to no in the interface object, set the validity-interval to several seconds longer than the default-update-interval. For example, if the default-update-interval is 5 seconds, set the validity-interval to 10 seconds. • If external-system-has-a-scheduler is set to yes in the interface object, set the value to the amount of time you are willing to use a retrieved value before it must be updated.
default-update-interval	The interval at which G2 should request a new value for the variable. The setting depends on whether exception reporting is enabled. See Exception Reporting. When exception reporting is enabled, the G2-PI Bridge automatically transmits changes when the value exceeds preset limits. Thus, the value of the default-update-interval is ignored; therefore, set the default-update-interval to none. When exception reporting is disabled, this value controls the rate at which the value is updated when external-system-has-a-scheduler is set to no. In this case, set the default-update-interval to the interval at which the value should be updated. If external-system-has-a-scheduler is set to yes, the validity-interval controls when the value is updated.
gsi-interface-name	The name of the GSI interface object that this variable will use to connect to the G2-PI Bridge. This name determines the variable's context and, therefore, whether or not the variable is updated by exception and its logging behavior.

Attribute	Description
gsi-variable-status	0: OK 128: The call to PI functions returned an error. 262146: Bad server. Reasons include the server does not exist, the server was not added to the server information file, communication with the server has failed, or the user name or password is incorrect. 262147: Bad value. Unable to decode digital state name. 262148: Bad type. Type conversion not supported. 262150: Missing PI variable definition. This error should not normally occur. The likely cause is lack of memory. 262151: No such tag. Request to access non-existent PI tag.
object-index	A numerical value identifying this variable object. This index is often referred to as the variable's **handle**. You use the object-index to identify the object in remote procedure calls. The G2-PI Bridge sets this value.
pi-point	The PI number of the referenced point. PI assigns a number to each point stored in its database. When a variable is first registered with the bridge, the bridge looks up the referenced point and stores its PI number in this attribute.
osipi-tagname	The reference to the PI point that supplies this variable with its value. The reference is of the form "tagname" when using the default server or "servername:tagname" when specifying a particular server. Note that double quotes are required. For details, see Referring to PI Tag Names.
osipi-data-type	Whether this variable is an integer (I), a real (R), or a digital (D). The value is determined by the osipi-int, osipi-real, or osipi-digital class definition. The bridge does not use this attribute; it is for your information only.
timed-value	The pi-integer-list or pi-quantity-list object that contains a history of the variable's values obtained from the PI system. pi-integer-list is used for osipi-int variables, and pi-quantity-list is used for osipi-real variables. You store the historical values in these attributes by calling get-time-vals or get-interp-vals. See History Operations.

Attribute	Description
timedate-text	The pi-text-list object that contains the PI system timestamps, as text, which corresponds to each of the values contained in the timed-value attribute. Each timestamp is of the form MMDDYYYYhhmmss. The time is expressed as a text string of month, day, year, hour, minute and second. Leading zeros are used to pad a number out to the required number of places. The twenty-four hour clock is used so hours range from 0 to 23. For instance, you can use this list to supply values for the time axis in a plot using timed-values.
timedate-seconds	The pi-quantity-list object that contains the PI system timestamps, in seconds, which corresponds to each of the values contained in the timed-values attribute, above. Each timestamp is a float.
last-recorded-pi-time	The PI system timestamp, as text, which corresponds to the current value for this variable. The timestamp is of the form MMDDYYYYhhmmss. Typically, the last-recorded-pi-time is the time at which the PI system obtained this value.

Referring to PI Tag Names

To access PI data from a particular server from within G2, you must precede the PI tag name with the server name and a colon. For example, to read the value of the PI point named `SINUSOID` from the server named `IE0`, you would set the osipi-tagname attribute of the G2 variable to:

```
IE0:SINUSOID
```

If you do not specify a server, the bridge attempts to read the data from the default server.

PI tag names sometimes contain a colon. If the portion of the tag name preceding the colon matches the name of a PI server, you *must* precede the tag name with the server name, even if you are reading data from the default server. For example, to read the value of the tag named `IE0:003` from the default server, named `PAU`, when `IE0` is also a PI server, you would set the appropriate attribute of the G2 variable to:

```
PAU:IE0:003
```

Creating Your Own PI Variable Classes

The two reasons you might want to create your own PI variable class instead of using one of the classes that are predefined in `g2-pi.kb` are:

- You need to retrieve PI attributes for the referenced PI point and store them in the variable.
- You need a variable of a type other than those that are predefined, that is, a symbolic or logical variable.

Rules for Defining Your Own PI Variable Class

In the tables that follow, Predefined Name refers to the corresponding attribute name in the predefined variable classes, Name Fixed? indicates whether the attribute name you specify must be the same as the predefined name or not, and Type is the G2 data type for the attribute.

The rules for defining your own PI variable class are:

- The class must inherit its definition from gsi-data-service and either a subclass of g2-variable or sensor.
- It should have these additional attributes:

Predefined Name	Name Fixed?	Type	Notes
osipi-tagname	no	text	The PI pointer. You may give this attribute whatever name you like. In prior documentation, this attribute was sometimes called itempath. The first attribute name in the identifying-attributes of corresponding interface objects should be the name you choose for this attribute.
object-index	yes	integer	
pi-point	yes	integer	
last-recorded-pi-time	yes	text	

- If you are using PI point attribute retrieval, you must have an attribute that can hold a structure and one attribute of an appropriate type for each PI tag attribute that you will retrieve. The name you choose for the structure attribute should be the second name you put in identifying-attributes of corresponding interface objects.

- If you will use RPCs to read historical values from PI archives, you must have the following 3 attributes:

Predefined Name	Name Fixed?	Type	Notes
timed-value	no	integer-list	If you use a name other than the predefined name for any of these three attributes, you must call rpc-define-history-attributes to tell the bridge where is should store retrieved historical values.
time-data-text	no	text-list	
time-data-seconds	no	quantity-list	

Defining a Variable Class that Inherits from OSIPI Variables

`g2-pi.kb` defines the class osipi-var, which is a quantitative variable, and osipi-digital, which is a text variable, both of which are pre-configured to retrieve current values and historical values.

Thus, if you need a quantitative, real, or integer variable that will hold PI tag attributes, you can define your variable class to inherit its definition from osipi-var. That way, you will only need to define the attributes required for retrieving PI tag attributes. In addition, because the attributes to hold historical values use the predefined names, you will not need to call rpc-define-history-attributes.

Similarly, if you need a text variable that will hold PI tag attributes, define its class to inherit its definition from osipi-digital.

Example

The following example shows how to define a PI variable that will hold a float or integer value and will store the PI tag's descriptor and high-limit values.

To define your own PI variable that stores PI attributes:

1. Merge `g2-pi.kb` into your application.
2. Create a new object definition by choosing KB Workspace > New Definition > Class Definition > Object Definition.
3. Configure the names attribute to be the name of your variable class.
4. Configure the direct-superior-classes to be osipi-var.

   Notice that the inherited-attributes of the definition updates to include all the attributes that are required for retrieving the value of the PI tag and for storing historical values.

5 Configure the class-specific-attributes as follows:

```
piattrs-spec initially is a structure (description: 101, hi-limit: 5018);
description initially is "";
hi-limit is given by a float-parameter, initially is given by a float-parameter
```

In this specification:

- piattrs-spec contains a structure that specifies the ID of the PI attributes to retrieve and the attributes of the variable in which to store the values.
- description is the name of the variable attribute that will hold the PI tag's descriptor attribute (101).
- hi-limit is the name of the variable attribute that will hold the high entry limit for this PI data point (5018).

For a list of PI attributes and their corresponding ID codes, see PI Point Attributes.

6 Create an instance of your variable class and place it on a workspace.

7 Create and configure an interface object whose identifying-attributes is set to:

```
osipi-tagname, piattrs-spec
```

8 Configure the attributes of the variable as follows:

Attribute	Value
osipi-tagname	A reference to a PI point that can be stored in a quantitative variable.
validity-interval	The length of time that the last recorded value should remain current.
default-update-interval	The interval at which G2 should request a new value for the variable.
gsi-interface-name	The name of the interface object created in step 7.

As soon as the variable is registered with the bridge, the object-index, pi-point, description, and hi-limit attributes will be updated.

PI Point Attributes

As the Example shows, you can define a variable class that retrieves PI point attributes and stores them in attributes of the variable. To summarize:

1 Define your own variable class with class-specific attributes for each PI attribute you want to retrieve, and a structure attribute that will be used to

specify the PI attributes to retrieve and the G2 variable attributes in which to store the values.

2 Configure the second attribute in the identifying-attributes of the interface object to be the structure attribute of your variable class.

3 Instantiate your variable class and configure its attributes: gsi-interface-name, osipi-tagname, default-update-interval, and validity-interval.

The bridge automatically retrieves the PI attributes when the variable is registered.

When configuring the structure, the name of each element is the name of an attribute of the G2 variable that will hold the value of one of the PI attributes. The value of each element of the structure is a code for the corresponding PI attribute to retrieve.

In the example, the structure was defined as follows:

```
structure(description:101,hi-limit:518)
```

The code for the PI attribute named Descriptor is 101, and code for the PI attribute named High Entry Limit is 518. The High Entry Limit is the sum of the Zero and Span PI attributes, which represents the maximum legal value for the PI point's value. Therefore, the structure stores the value of the Descriptor PI attribute in the G2 variable attribute named description, and it stores the maximum legal value in the G2 variable attribute named hi-limit.

Normally, PI attributes do not change; they represent static information about the point. As a result, the bridge only reads their values when a variable is registered. If you do change a tag's attributes in PI after they have been read by G2, the values stored in the G2 variable will not be updated. However, the new values will be read next time the variable is registered. You can force re-registration by disabling and then re-enabling the variable or by restarting G2.

The codes for the PI attributes are:

ID	PI attribute	Type	Notes
1	Canonical Data Type	integer	
100	Engineering Units	text	e.g., "kg/s", "*C"
101	Descriptor	text	e.g., "Evaporator 6 Temp"
5004	Stepped	logical	
5005	Archiving	logical	

ID	PI attribute	Type	Notes
5006	Extended Descriptor	text	
5009	Instrument Tag	text	
5010	Point Source	text	
5013	Tag	text	
5014	Exception Max Interval	integer	seconds
5015	Exception Min Interval	integer	seconds
5016	Exception Deviation	float	
5017	Filter Time Constant	integer	seconds
5018	High Entry Limit	float	zero + span
5019	Low Entry Limit	float	zero
5020	Typical Value	float	
5200	Display Digits	integer	
6004	Scanning	logical	
6005	Compressing	logical	
6014	Compression Max Interval	integer	seconds
6015	Compression Min Interval	integer	seconds
6016	Compression Deviation	float	
7000	Point ID Number	integer	
7001	Creation Date	float	
7002	Creator	text	
7003	Change Date	float	

ID	PI attribute	Type	Notes
7004	Changer	text	
7005	Location Parameter #1	integer	
7006	Location Parameter #2	integer	
7007	Location Parameter #3	integer	
7008	Location Parameter #4	integer	
7009	Location Parameter #5	integer	
8000	Source Tag	text	
8001	Square Root	integer	
8002	Totalization Code	integer	
8003	Conversion Factor	float	

How the Bridge Converts Data

Generally, the type of variable you use to retrieve a value from a PI point matches the point's type. For example, if you are reading a `float32` value, you would probably use a float or quantitative variable to hold the value. However, in most cases where it makes sense to do so, the PI bridge performs type conversion if the G2 variable and PI point types do not match.

The following table shows how the G2-PI bridge converts the data. Note that not all combinations of data types are supported.

		G2			
		Integer	**Float**	**Text or Symbol**	**Logical**
PI	**Integer**	copy	convert to float	convert to text	0=true; others=false
	Float	round	copy	convert to text	0.0=true; others=false
	Digital			tag state	
	Text			copy	
	Time		PI time	MMDDYYHHmmSS	

If a conversion is not supported, a value of 262148 is stored as the variable's status.

Exception Reporting

In general, PI variables update their data based on the default-update-interval or validity-interval of the variable. In these cases, it is G2 that requests new values for variables. However, the G2-PI Bridge provides another way of updating variable values known as exception reporting.

When exception reporting is enabled, it is the bridge that informs G2 that the value of a variable should be changed. This occurs when the value of the corresponding PI point changes by more than limits defined within the PI point. The rules for when this occur are defined by PI and involve several attributes of the PI point including `ExcDev`, `ExcMin`, and `ExcMax`. The person who configures the PI point is responsible for setting these attributes. For more information, see OSIsoft's documentation on data flow.

Two reasons you might want to use exception reporting are:

- Your system is not responsive because you are updating a large number of variables with short default update times. Using exception reporting could significantly improve performance, especially if the values of the PI points are changing slowly.
- You want to minimize the delay between the time the value of a PI point changes and the time it is reported to G2. Although it depends upon such factors as program and network load, the change of the value of a PI point by more than the preset amount will typically be reported within a few seconds.

For information on configuring exception reporting, see Exception Report Configuration.

Registering Variables

G2 is responsible for informing the G2-PI Bridge about any variables that should receive their values from PI. This process is known as registering the variable.

When the variable is registered, the bridge:

- Uses the first identifying attribute to find the PI point that will be the source of the variable's value.
- Stores the variable's handle and the PI point number in variable's object-index and pi-point attributes.
- Retrieves any requested PI attributes and stores them in the variable.
- Saves information that will be needed to quickly process future requests for values and to process exception reports.

G2 registers a variable, if the variable has not yet been registered and if one of the following conditions occurs:

- It has a default-update-interval other than none and G2 is started or an attribute of the variable is changed.
- G2 requests the value of the variable. Examples are: a collect data statement is executed, the value is needed to evaluate a rule, or a readout table that displays the value is updated.
- An update action is executed.
- There is a request to set the variable's value.
- A call to the g2-register-on-network system procedure is executed for the variable.

If you create variables that should be updated by exception reporting, you should set their default-update-interval to none and their validity-interval to none. If you configure them and then watch their tables, you will see that they are not updated. The reason is that they were never registered.

The solution is either to actively use their values from within G2 or to use the g2-register-on-network system procedure. For example, the following procedure registers every variable on a workspace:

```
regall()
l : class variable ;
handle: integer ;
begin
    if the gsi-interface-status of XIO = 2 then
        begin
            for l = each variable upon this workspace do
                handle = call g2-register-on-network(l, xio)
        end
end
```

The g2-register-on-network system procedure is defined in `sys-mod.kb`, a knowledge base that is delivered with G2. You need to merge this file with your application to use g2-register-on-network. For more information, see the *G2 System Procedures Reference Manual*.

Retrieving Historical Values

You might need to keep a record of the values of a particular variable over some period time. Typically, in G2 applications, you keep a history by configuring the history-keeping-spec of the variable to keep a history. However, for values that originate from the PI system, keeping a history in G2 would be redundant and would waste storage space, because the PI system already maintains a record of each variable's history.

Thus, the G2-PI Bridge uses a remote procedure call to read history values from PI. The results are stored in lists, which are attributes of a PI variable. If you use the classes that are predefined in `g2-pi.kb`, these attributes are already defined for you.

As described in Using OSIPI Variables, each PI variable contains a reference to a PI point. To retrieve values from that point's history, you call one of these two RPCs:

- get-time-vals reads historical values.
- get-interp-vals retrieves interpolated values.

In either case, you pass to the RPC a handle to the corresponding PI variable.

The results are stored in a list that is an attribute of the PI variable. The times associated with the values are stored in two different formats in lists that are also attributes of the variable.

Preparing to Retrieve Historical Values

To retrieve historical values, your PI variable class must define three list attributes of the proper types. If you use the predefined variable classes, these attributes are predefined.

The attributes of the variable are:

- The results list, which is named timed-value in the predefined PI variable classes. The type of this list depends upon the PI point type. For example, if you are reading the history of an integer, the attribute should be of type integer-list.

  When you use a predefined PI variable class, this attribute will be of type pi-quantity-list, pi-integer-list, or pi-text-list, depending upon which of the predefined classes you are using. Except for their names, these types are identical to quantity-list, integer-list, and text-list, respectively. The advantage of using these alternate names is that it makes it clear to the user that the attributes of these types are intended to hold values to be retrieved by PI.

- The PI time list, which is named timedate-seconds in the predefined PI variable classes. The list is of type pi-quantity-list, which is the same as quantity-list.

  Each element of this list is the time in PI format (a floating point number) that the value in the corresponding position of the results list was recorded.

- The time as text list, which is named timedate-text in the predefined PI variable classes. The list is of type pi-text-list, which is the same as text-list.

  Each element of this list is a string of the format "MMDDYYYYhhmmss", which represents the time the corresponding value of the results list was recorded.

Therefore, before you can retrieve history values for a PI point, you must define the required list attributes in your PI variable class. However, if you use one of the predefined classes, these attributes are already defined.

If you define your own variable classes, you may use any name for these list attributes. If you use names other than the defaults, then you must tell the bridge the names that you used so it knows where it should store the retrieved values. You do this by calling rpc-define-history-attributes, whose signature is:

rpc-define-history-attributes
("TIMEVECTOR", *name-of-value-list-attribute*:text,
name-of-pi-time-list-attribute:text, *name-of-time-as-text-list-attribute*:text)
across *name-of-interface-object*
-> *status*: integer, *message*: text

For example:

```
c: integer ;
m : text ;
begin
    c, m = call rpc-define-history-attributes("TIMEVECTOR", “histvals”,
        “flttimes”, “txttimes”) across piio ;
```

where:

TIMEVECTOR is a constant and should appear exactly as shown.

histvals is the name of the attribute that will hold the retrieved values, where the type depends on the PI point type.

flttimes is the name of the quantity-list or float-list that will hold the PI times.

txttimes is the name of the text-list that will hold the text strings representing the time the retrieved values were recorded.

piio is the name of an interface object connected to the G2-PI Bridge.

Retrieving Historical Values

The two RPCs you can use to retrieve values from the PI historian are:

- get-time-vals retrieves specific values.
- get-interp-vals retrieves interpolated values.

They have the following signatures:

get-time-vals
(*handle-of-pi-variable*: integer, *start-time*: text, *end-time*: text, *number-of-points*: integer)
-> *status*: integer, *message*: text

get-interp-vals
(*handle-of-pi-variable*: integer, *start-time*: text, *end-time*: text, *number-of-points*: integer)
-> *status*: integer, *message*: text

where:

handle-of-pi-variable is the object-index attribute of a PI variable, which is set by the bridge to the variable’s handle.

start-time and *end-time* are of the form "MMDDYYYYhhmmss".

number-of-points is the number of points to retrieve.

For information on the return values, see the description of these RPCs in History Operations.

For example:

```
c: integer;
m: text ;
begin
    c, m = call get-time-vals(the object-index of PIVAR3, “07042003000000”,
        “07052003000000”, 25) ;
```

This call passes a request to PI to retrieve from the PI historian 25 values recorded between midnight July 4, 2003 and midnight July 5, 2003 for the PI point that is referenced by the G2 variable PIVAR3. Assuming that PIVAR3 uses the default names for the history attributes, the results will be stored in the timed-value, timedate-seconds, and timedate-text attributes of PIVAR3.

The decision of which points to retrieve is made by PI, not by the bridge. Likewise, when requesting interpolated values, it is PI that calculates the times and the values. The bridge simply acts as an intermediary.

Writing to PI

To write a value to a PI point, use the G2 set action. The format is:

set *pi-variable* to *value*

When you execute this command, G2 attempts to write the specified value to the PI point referenced by the specified variable. If the attempt fails, the bridge logs an error message. When using this syntax, the value is stamped by the server with the current time.

You can optionally specify the timestamp that should be used with the `SET` command. To do this, set the point to a sequence instead of a single value. The first element of the sequence is the value to which the point is being set. The second is the timestamp in the date format defined by OSIsoft.

For example, the following code sets the value of the point named `PI` to 0.50 with the specified timestamp:

```
SET PI TO sequence(0.50, "09-NOV-2005 14:09:02")
```

See Appendix B of OSIsoft’s *PI UDS Reference Guide* for details.

The PI server rejects any attempt to use a time in the future.

5

Remote Procedure Calls (RPCs)

Describes the Remote Procedure Calls (RPCs) for the G2-PI Bridge.

Introduction

This section describes the remote procedures published by the G2-PI Bridge. Each entry uses the notation as listed below.

General Operations

shutdown

( )

Causes the bridge to shut down once all contexts are disconnected.

stop-bridge

(*stop*: truth-value)

This procedure gives a way to abort the shutdown process. When the argument is false, the bridge cancels any shutdown requests that were waiting for all contexts to disconnect. When the argument is true, the procedure behaves just like shutdown.

Item Operations

read

(all remaining item-or-value as handle)

Performs a read from device for the specified PI variables. This procedure is primarily used for diagnostics or for particularly critical operations.

Argument	Description
all remaining item-or-value as handle	The PI variables passed by reference whose data should be read. The handle can be found in the variable's object-index attribute.

translate-status

(class *variable* as handle)
-> *status*: text

Returns a text description of the status of the PI variables.

Argument	Description
class *variable* as handle	The PI variable, passed by reference, whose status should be translated. The handle can be found in the variable's object-index attribute.

Return Value	Description
status	Description of variable status.

get-property

(class *variable* as handle, *id*: integer)
-> *property*: value

Returns the value of a specified PI attribute associated with the specified variable.

Argument	Description
class *variable* as handle	The handle of a PI variable that refers to the PI point in which you are interested. The handle can be found in the variable's object-index attribute.
id	The code for the PI attribute to retrieve. These are the same codes that are used for automatic retrieval of PI attributes. See PI Point Attributes.

Return Value	Description
value	The value of the specified PI point attribute.

For a list of values for the property ID numbers, see PI Point Attributes.

History Operations

rpc-define-history-attributes

(*vector-class* text, *values-list-attribute*: text, *timestamps-attribute*: text, *text-timestamps-attribute*: text)
-> *status*: text, *message*: text

Defines the attribute names used by get-time-vals and get-interp-vals to hold historical information of a variable's values. You must call rpc-define-history-attributes if the attributes you use to store historical values have names other than the default. See Retrieving Historical Values

Argument	Description
vector-class	The name of a specific class of vectors. In this version of the bridge, use only the class timevector.
values-list-attribute	The name of the attribute that will be used to store a list of values. The type of this attribute depends upon the type of the PI point. For example, if the PI point is an integer, this attribute should be an integer-list or equivalent.
timestamps-attribute	The name of the attribute that will be used to store a list of numeric timestamp values. The list will be a quantity-list or equivalent.
text-timestamps-attribute	The name of the attribute that will be used to store a list of timestamps in text form. The list will be a text-list or equivalent.

Return Value	Description
status	The status code. It is equal to 1 if the function completes successfully. Negative values indicate an error.
message	A descriptive status message.

Example:

```
code, msg = call rpc-define-history-attributes
    ("timevector", "timed-value", "timedate-seconds", "timedate-text")
    across osipi-one;
```

get-time-vals

(*handle-of-pi-variable*: integer, *start-time*: text, *end-time*: text, *number-of-points*: integer)
-> *status*: text, *message*: text

Requests a list of past values from the PI system for a PI variable. If the attributes that will hold the retrieved values have names other than the default, the call to rpc-define-history-attributes, which identifies the attributes to receive the values, must have been made previously during the current G2 session. The values are stored in a list attribute of the variable.

Argument	Description
handle-of-pi-variable	Identifies the handle of a PI variable that refers to a PI point. You obtain this handle by referring to the object-index of the PI variable.
start-time	Specifies the starting PI system time for the requested values. Time is expressed as a text string of the form MMDDYYYYhhmmss.
end-time	Specifies the ending PI system time for the requested values. Time is expressed as above.
number-of-points	Specifies the number of values desired.

Return Value	Description
status	The status code. It is equal to 1 if the function completes successfully. Negative values indicate an error.
message	A descriptive status message.

The following call requests 100 values of the variable *object-name* over the period starting on January 1, 2003 at 9:15pm and ending on January 2, 2003 at 9:15pm from the PI system identified by the osipi_interface object osipi-one.

```
code, msg = call get-time-vals
(the object-index of osipi-one, ”01012003211500”, “01022003211500”,
    100)
across osipi-one;
```

get-interp-vals

(*handle-of-pi-variable*: integer, *start-time*: text, *end-time*: text, *number-of-points*: integer)
-> *status*: text, *message*: text

Requests a list of past values from the PI system for PI variable. The values returned are interpolated values, not actual stored PI data. They are spaced evenly within the time range given.

If the attributes that will hold the retrieved values have names other than the default, the call to rpc-define-history-attributes, identifying the attributes that are to receive the values, must have been made previously during the current G2 session.

Argument	Description
handle-of-pi-variable	Identifies the handle of a PI variable that refers to a PI point. You obtain this handle by referring to the object-index of the PI variable.
start-time	Specifies the starting PI system time for the requested values. Time is expressed as a text string of the form MMDDYYYYhhmmss.
end-time	Specifies the ending PI system time for the requested values. Time is expressed as above.
number-of-points	Specifies the number of values desired.

Return Value	Description
status	The status code. It is equal to 1 if the function completes successfully. Negative values indicate an error.
message	A descriptive status message.

The following call requests 100 interpolated values of the variable *object-name* over the period starting on January 1, 2003 at 9:15pm and ending on January 2, 2003 at 9:15pm from the PI system identified by the osipi_interface object osipi-one.

```
code, msg = call get-interp-vals
(the object-index of object-name, ”01012003211500”, “01022003211500”,
100) across osipi-one;
```

get-pitime

( )

-> *status*: text, *message*: text, *time*: text, *year*: integer, *month*: integer, *day*: integer, *hour*: integer, *minute*: integer, *second*: integer,

Requests the current PI system time from the default server. You can use this RPC to determine the amount of skew between the G2 time and the default PI system time. No arguments are required.

Return Value	Description
status	The status code. It is equal to 1 if the function completes successfully. Negative values indicate an error.
message	A descriptive status message.
time	The current PI system time, expressed as a text string of the form MMDDYYYYhhmmss.
year, *month*, *day*, *hour*, *minute*, *second*	The current PI system time, expressed as a series of integers.

Example:

```
code, msg, time, year, month, day, hour, minute, second =
    call get-pitime( ) across osipi_one
```

Logging Operations

The G2-PI Bridge can report various information. The information that is reported depends upon the “log level” you select. A log level of 1 causes only errors to be reported. Log level 2 causes errors and warnings to be reported. Higher log levels may report internal information, which can help Gensym customer support evaluate unexpected behavior.

You have many choices for logging information. You can report messages to the screen, write them to a file, send them to the G2 Message Board, send them to a

G2 procedure for processing, write them to the PI log, or any combination of these options.

For each interface object that is connected to the G2-PI Bridge, you have what is called a **context**. You may have up to five contexts in a G2-PI Bridge application. Configuration of most logging options is on a per-context basis. This feature gives you the ability to partition your set of variables and configure the amount of information that will be reported for each subset and how that information will be reported.

If you do nothing to change the configuration, the default log level is 2 and the default log destinations are the screen and the G2 procedure osi-error. However, you can also start the bridge with command-line options that change the default behavior, as described in Command-Line Options. These defaults apply across all contexts.

Your next level of control is to use initialization string options to set the logging behavior for individual contexts. For more information, see Remote-process-initialization-string.

Your final level of control is to use RPCs. You can use them to change the behavior of individual contexts back and forth, using whatever criteria are appropriate for your situation. This feature gives you the ability to selectively log information.

Logging to a G2 Procedure

If you are going to use the option to log to a G2 procedure, which happens by default, you must define the G2 procedure osi-error in your knowledge base. The procedure must have the following signature:

```
osi-error
    (error-code: integer, error: text)
    -> (return: integer).
```

When a logging event occurs and log-to-g2 is enabled, the bridge calls the osi-error procedure in G2. If the logging event returns an error code, the code is passed as the first argument to the procedure. The message that is being logged is passed as the second argument. The bridge does not use the return value from the procedure.

`g2-pi.kb` provides you with a sample osi-error procedure, which performs no action. If you merged this KB into your application, you can modify the procedure to respond to logged events.

RPCs for Logging

log-force-writes

(*active*: truth-value)
-> (*success*: integer)

Normally, when a log message is written to a file, it is buffered in memory and will be written to disk when the computer is not busy with other activities. In this case, if some unexpected behavior causes the bridge to terminate, it can happen that messages never get written to the disk.

If you call log-force-writes(true), log messages will not be buffered. Instead, they will be written immediately to disk before processing by the bridge continues.

This is the only log configuration RPC that applies across all contexts.

Argument	Description
active	True to activate forced writes, otherwise false.

Return Value	Description
success	Zero if the call is successful, otherwise -1.

log-level

(*level*: integer)
-> (*success*: integer)

Changes the log level.

Argument	Description
level	The new log level (0 – 9).

Return Value	Description
success	Zero if the call is successful, otherwise -1.

log-message

(*message*: text, *timestamp*: truth-value)
-> (*success*: integer)

Gives the user the ability to add optionally time-stamped messages to the log.

Argument	Description
message	The message text to write to the log.
timestamp	True to timestamp the log message, otherwise false.

Return Value	Description
success	Zero if the call is successful, otherwise -1.

log-to-file

(*active*: truth-value)
-> (*success*: integer)

Controls whether log messages are output to a file.

Argument	Description
active	True to write log messages in a file, otherwise false.

Return Value	Description
success	Zero if the call is successful, otherwise -1.

log-to-g2
(*active*: truth-value)
-> (*success*: integer)

Controls whether log messages are sent to the osi-error procedure, as described at the beginning of this section.

Argument	Description
active	True to send log messages to the osi-error procedure, otherwise false.

Return Value	Description
success	Zero if the call is successful, otherwise -1.

log-to-message-board
(*active*: truth-value)
-> (*success*: integer)

Controls whether log messages are displayed in G2 on the Message Board.

Argument	Description
active	True to display log messages in G2 on the Message Board, otherwise false.

Return Value	Description
success	Zero if the call is successful, otherwise -1.

log-to-pi

(*active*: truth-value)
-> (*success*: integer)

Controls whether log messages are written in the PI log file.

Argument	Description
active	True to write log messages in the PI log file, otherwise false.

Return Value	Description
success	Zero if the call is successful, otherwise -1.

log-to-screen

(*active*: truth-value)
-> (*success*: integer)

Controls whether log messages are output to the console window.

Argument	Description
active	True to display log messages in the console window, otherwise false.

Return Value	Description
success	Zero if the call is successful, otherwise -1.

log-settings

( )
-> (*file-name*: text, *log-level*: integer, *max-size*: integer, *force*: truth-value, *log-to-screen*: truth-value, *log-to-message-board*: truth-value)

Retrieves the state of the logging options that were available in Version 4.0 of the bridge.

Return Value	Description
file-name	The log file name.
log-level	The current log level.
max-size	The maximum log file size, in bytes. A value of 0 means that the log file size is limited only by available disk space.
force	True if forced writes are active, otherwise false.
log-to-screen	True if messages are output to console window, otherwise false.
log-to-message-board	True if messages are displayed in the G2 Message Board, otherwise false.

log-settings2

( )
-> (*file-name*: text, *log-level*: integer, *max-size*: integer, *force*: truth-value, *log-to-screen*: truth-value, *log-to-message-board*: truth-value, *log-to-g2*: truth-value, *log-to-pi*: truth-value)

Retrieves the current log settings for all logging options, including those that did not exist in earlier versions of the bridge.

Return Value	Description
file-name	The log file name.
log-level	The current log level.
max-size	The maximum log file size, in bytes. A value of 0 means that the log file size is limited only by available disk space.
force	True if forced writes are active, otherwise false.
log-to-screen	True if messages are output to console window, otherwise false.

Return Value	Description
log-to-message-board	True if messages are displayed in the G2 Message Board, otherwise false.
log-to-g2	True if messages are sent to the osi-error procedure, otherwise false.
log-to-pi	True if messages are displayed in PI, otherwise false.

Index

H

I

L

M

N

O

P

R

S

T

U

V

W

X