BridgeVIEW

VI Server Development Toolkit Reference Manual

March 1997 Edition
Part Number 321297A-01

Internet Support

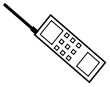
support@natinst.com
E-mail: info@natinst.com
FTP Site: ftp.natinst.com
Web Address: http://www.natinst.com

Bulletin Board Support

BBS United States: (512) 794-5422
BBS United Kingdom: 01635 551422
BBS France: 01 48 65 15 59

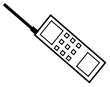
Fax-on-Demand Support

(512) 418-1111



Telephone Support (U.S.)

Tel: (512) 795-8248
Fax: (512) 794-5678



International Offices

Australia 02 9874 4100, Austria 0662 45 79 90 0, Belgium 02 757 00 20,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, Denmark 45 76 26 00,
Finland 09 527 2321, France 01 48 14 24 24, Germany 089 741 31 30, Hong Kong 2645 3186,
Israel 03 5734815, Italy 02 413091, Japan 03 5472 2970, Korea 02 596 7456,
Mexico 5 520 2635, Netherlands 0348 433466, Norway 32 84 84 00, Singapore 2265886,
Spain 91 640 0085, Sweden 08 730 49 70, Switzerland 056 200 51 51, Taiwan 02 377 1200,
U.K. 01635 523545

National Instruments Corporate Headquarters

6504 Bridge Point Parkway Austin, TX 78730-5039 Tel: (512) 794-0100

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

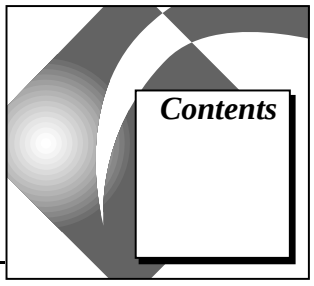
EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.



About This Manual

Chapter 1
VI-Based Server Interface to the BridgeVIEW Engine

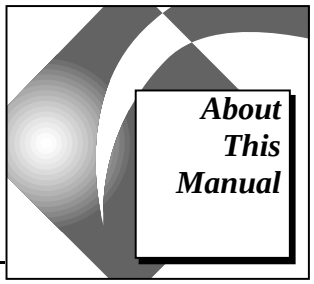
Chapter 2
Function Reference

Appendix
Customer Communication

Glossary

Figures

Tables



The *VI Server Development Toolkit Reference Manual* describes the use of VI-based servers with the BridgeVIEW Engine. This document contains descriptions and examples of the VIs used to register and execute the VI-based servers.

To use this document effectively, you should be familiar with programming in G. We also recommend that you review the following chapters in the *BridgeVIEW User Manual*:

- Chapter 1, *Introduction*
- Chapter 2, *BridgeVIEW Environment*
- Chapter 8, *Industrial Automation Device Servers*

Organization of This Document

The *VI Server Development Toolkit Reference Manual* is organized as follows:

- Chapter 1, *VI-Based Server Interface to the BridgeVIEW Engine*, describes the use of G or VI-based servers with the BridgeVIEW Engine. This chapter also describes how to configure and register a VI-based server, how a server operates, and explains error handling and performance issues you might encounter.
- Chapter 2, *Function Reference*, describes the VIs that register VI-based servers and interface the VI-based servers to the BridgeVIEW Engine during server execution.
- Appendix, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.

- The *Glossary* contains an alphabetical list and description of terms used in this document, including abbreviations, acronyms, metric prefixes, mnemonics, and symbols.

☞ **Note:** ***You can view this file using any version of Adobe Acrobat Reader. For best results when printing this file, use Adobe Acrobat Reader 3.0. Adobe Acrobat Reader 3.0 is available from the Adobe web site at*** `http://www.adobe.com/acrobat`**.**

Conventions Used in This Document

The following conventions are used in this document:

bold	Bold text denotes VI parameters, menus, menu items, or dialog box buttons or options.
italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept.
bold italic	Bold italic text denotes a note.
`monospace`	Lowercase text in this font denotes text or characters that are to be literally input from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, variables, filenames, and extensions, and for statements and comments taken from program code.
<>	Angle brackets enclose the name of a key on the keyboard—for example, <PageDown>.
-	A hyphen between two or more key names enclosed in angle brackets denotes that you should simultaneously press the named keys—for example, <Control-Alt-Delete>.
»	The symbol » leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options»Substitute Fonts** directs you to pull down the **File** menu, select the **Page Setup** item, select **Options**, and finally select the **Substitute Fonts** option from the last dialog box.
<Control>	Key names are capitalized.

paths	paths in this document are denoted using backslashes (\) to separate drive names, directories, and files, as in `drivename\dirlname\dir2name\myfiles.`
☞	This icon to the left of bold italicized text denotes a note, which alerts you to important information.

Related Documentation

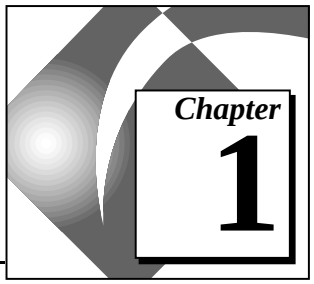
The following documents contain information that you might find helpful as you read this document:

- *BridgeVIEW User Manual*
- *BridgeVIEW Device Server Toolkit Reference Manual*
- *G Programming Reference Manual*

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this document contains comment and configuration forms for you to complete. These forms are in Appendix, *Customer Communication*, at the end of this document.

VI-Based Server Interface to the BridgeVIEW Engine



This chapter describes the use of G or VI-based servers with the BridgeVIEW Engine. This chapter also describes how to configure and register a VI-based server, how a server operates, and explains error handling and performance issues you might encounter. Finally, this chapter includes an example of a VI-based server design.

Why Develop VI-Based Servers?

The BridgeVIEW Engine can interface with any device server that uses the BridgeVIEW Engine Server interface. While the server does not need to be implemented in G, it must use G to interface with the BridgeVIEW Engine using the BridgeVIEW Engine Server Interface VIs.

☞ **Note:** ***The two exceptions for using G to interface with the BridgeVIEW Engine are as follows:***

1. ***If a server is written as a DLL to the National Instruments Industrial Automation Device Server Specification (IA Device Server), the BridgeVIEW Engine can interface to it through the IAIO Server Proxy.***
2. ***If a server is implemented as a Windows DDE server, the BridgeVIEW Engine can interface to it using the DDE Server Proxy.***

For medium-experienced G programmers, developing and using a VI-based server is simple. Some reasons to develop a VI-based server include the following:

- Use existing G-based applications as a server.
- Simulate hardware or an actual system.
- Complete simple, yet specific tasks.
- Use your G programming skills.

How Does a VI-Based Server Work?

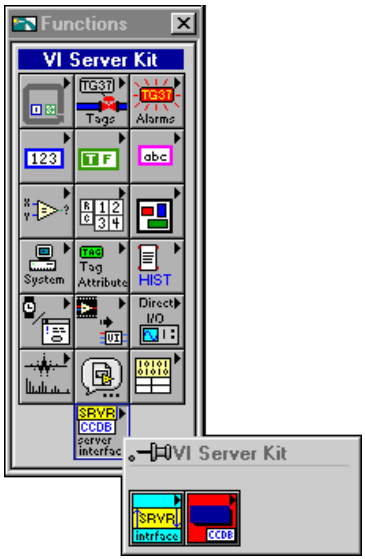
The server supplies data points from several input items to the BridgeVIEW Engine as these points are read. The BridgeVIEW Engine also can send values for output items. BridgeVIEW uses queues to communicate between the BridgeVIEW Engine and its servers or server proxies.

The BridgeVIEW Engine accepts double values as well as string data, a packed array of unsigned 8-bit integers, from servers. Double values can be interpreted as analog, discrete (Boolean), or bit array (bit vectors up to 32-bits in length), depending on your BridgeVIEW tag configuration for a specific device item. All scalar values must be converted to double floating-points to pass to the BridgeVIEW Engine. The server must convert correctly signed or unsigned values to double floating-point representations.

Ideally, the server time stamps values as they are acquired from items, recording the time at which the value was acquired or sampled. The **timestamp** is in seconds since January, 1904, (Universal time) and is a double floating-point number rather than an unsigned 32-bit integer. Therefore, resolution is less than 1 second. If the server cannot time stamp the values as they are acquired, the server can set a flag so the BridgeVIEW Engine time stamps the value when it is received.

What Does My Toolkit Include?

Once you install the VI Sever Development Toolkit, a new palette is added to your BridgeVIEW **Functions** palette. This new **VI Server Kit** palette, shown below, contains the VIs for VI Server Interface and VI Server Registration. A **VI Server Typedefs** palette also is added to your BridgeVIEW **Controls** palette.



The VI Server Development Toolkit installs some examples in the `BridgeVIEW\_servers\Development\VI Server\Sample` directory to help you get started. These examples consist of several server shells, an example server registration VI, and an example server configuration utility VI.

BridgeVIEW also includes the simulation servers, SIM Server and Tanks Server, in the `_servers` directory. These simulation servers are complete server implementations in source code form.

The VI Server Development Toolkit adds the Interactive Server Tester utility to the **Server Tools** section of the BridgeVIEW **Projects** menu. You can use this utility to simulate execution of a VI-based server in the engine environment. Use the Help window (<Ctrl-H>) when you run this utility for more information on how to operate the utility.

VI-Based Server Development Tools

Several VI-based server examples ship with BridgeVIEW. These are good examples to experiment with to become familiar with developing servers. The VI Server Development Toolkit also contains the example VIs shown in this document.

You can use the BridgeVIEW Server Browser utility to view the devices, items, and capabilities registered by a server interactively. This utility can launch a Server Configuration Utility (if available) or it can be used from the BridgeVIEW Engine to display the front panel of a server while it is running.

You cannot debug a server while it is running in the BridgeVIEW Engine process. Use the Interactive Server Tester instead. This tool emulates the BridgeVIEW Engine/Server Interface in the BridgeVIEW user process. With this tool, you also can test launching, error reporting, and reading and writing server queues in a full G development system environment. This tool is included in the VI Server Development Toolkit and installed under the **Projects»Server Tools** menu.

Server-User Interface

The front panel of the server VI remains hidden from the user during VI execution, but can be displayed using the Server Browser utility from the BridgeVIEW Engine. In addition, the server can display general server status or other information on its front panel, if desired. The user sees this information only if the front panel is open. Use **VI Setup** to hide the server toolbar and prevent the user from closing or aborting the server while it is running.

Viewing and Printing This File

You can view this file using any version of Adobe Acrobat Reader. For best results when printing this file, use Adobe Acrobat Reader 3.0. Adobe Acrobat Reader 3.0 is available from the Adobe web site at `http://www.adobe.com/acrobat`.

Server Configuration

Typically, a server has a configuration utility associated with it with which you can do the following:

- Set up communication parameters
- Specify how to handle errors
- Configure hardware
- Configure poll rates
- Define a set of valid device and item names (optional)

The user executes this utility before using the BridgeVIEW Tag Configuration Editor to configure any tags using the server and before the BridgeVIEW Engine executes the server.

During configuration, the server must register information about itself and the devices and items it manages with BridgeVIEW. While servers are not required to have configuration utilities, they must be registered before BridgeVIEW can use them.

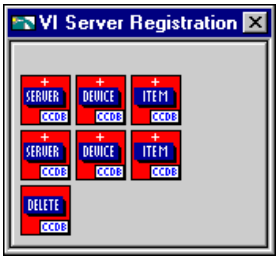
Server Registration

BridgeVIEW uses the Common Configuration Database (`.ccdb`) file to determine what servers are present and details about those servers, such as the following:

- Paths
- Registered devices
- Registered items

The CCDB manages the registered BridgeVIEW Server information. This database maintains tables of servers, devices, and items. The more information a server registers, the easier it is to use with BridgeVIEW because the user can configure and verify a tag more efficiently. You typically register information about your server as part of your Server Configuration Utility. If you do not have a configuration utility for your server, you must provide a VI that performs the registration.

The server uses a set of subVIs to communicate with the BridgeVIEW Engine. These subVIs are contained in the **VI Server Registration** palette shown below.



You can register server information from a DLL, C program, or any programming language that can call a DLL or use OLE automation. If your server configuration utility or program that registers the server is not written in G, see the *BridgeVIEW Device Server Toolkit Reference Manual* to learn how to register your server from a C/C++ program.

To register your server for use with BridgeVIEW, use the SVRG Add Server Row VI. This VI creates an entry for your server in the Servers table of the CCDB. When you register your server using the SVRG Add Server Row VI, it appears in the list of servers accessible from the BridgeVIEW Tag Configuration Editor Server list with the name supplied in **Server Name**.

At the very minimum, a server must register the following information to be usable from BridgeVIEW:

- **Server Name**
- **Server Type**—VI, IAIO
- **Server Launch Path** to VI or executable

☞ **Note:** ***The* Server Name *is the same name as that used by the server in the block diagram when using the Server Interface (SRVR) VIs.***

***For VI-based servers,* Server Type *must be* VI.**

Additional information a server can register includes the following:

- Predefined device names
- Predefined item names
- For each item name, item information:
  - **item data type**
  - allowed item directions (**access rights**): input, output, I/O (required)
  - **item range max and min** (optional)
  - **item unit** (optional)
  - **item max length** (optional)

You might want to register one or more devices recognized by your server or configured as part of your server configuration. Use the SVRG Add Device Row VI to register a device for your server with BridgeVIEW. This VI creates an entry for your device in the Devices table of the CCDB. If your server can interpret device strings to identify a device, you are not required to register the device; however, doing so makes it easier for the user to select a device.

When you register one or more devices for a server, the device name appears in the Device list when that server is selected in the BridgeVIEW Tag Configuration Editor. Even if you do not have a specific device or all items of interest are associated with a single device, you must register the device if you plan to register any items. In this case, use a default device name such as ALL.

You also might want to register one or more items recognized by your server for a specific device or configured as part of your server device configuration. Use the SVRG Add Item Row VI to register an item for your server device with BridgeVIEW. You must register a device before you can register an item for that device. This VI creates an entry for your item in the Items table of the CCDB. If your server is able to interpret item strings to identify a device item, you are not required to register the item; however, doing so makes it easier for the user to select an item. When you register one or more items for a server device, the item name appears in the Item list when that server and device are selected in the BridgeVIEW Tag Configuration Editor.

Registering engineering unit information is optional and only should be done if the actual engineering range and unit information for the item can be predetermined. If you do not register engineering unit information, the user can enter the information using BridgeVIEW.

Use the SVRG Delete Row VI to delete a specific row from the Server, Device, or Item tables. If you delete a server from the Server table, all devices for that server in the Device table and all items for that server in the Items table are deleted automatically. You do not need to delete devices and items individually if you want to delete them all. Similarly, if a device is deleted from the Devices table, all items for that device in the Items table are deleted automatically.

The following VIs query information once it is registered in the CCDB:

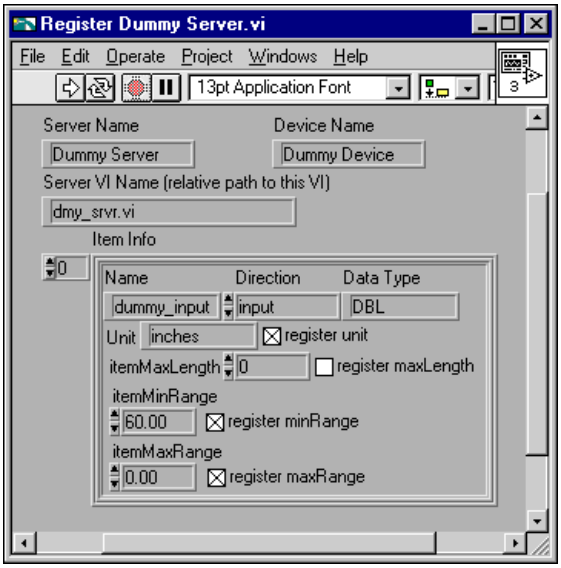
- SVRG Get Server Row VI
- SVRG Get Device Row VI
- SVRG Get Item Row VI

You can use these VIs if you save information in the CCDB that is useful for your server at launch time. You also can use them to see whether your information is registered successfully.

Register Server Example

The Register Dummy Server VI, shown in Figure 1-1, illustrates how to register information for your server. The user can configure server behavior, devices, or server communications channels with the configuration utility. Registering server, device, and item information is part of server configuration. If you develop a VI-based configuration utility, include the server registration as part of it. In other cases, such as a simple device or fixed server configuration, or if you are writing a server to simulate tags, you might not develop a server configuration utility. In these cases, you must develop a VI similar to the Register Dummy Server VI in Figure 1-1, and register the items for which your server generates or accepts data. Most of the simulation server examples for BridgeVIEW have a register server VI similar to the one shown in Figure 1-1.

The `servers/Development/VI Server/Sample` folder includes this VI and a more complete server configuration utility VI example.



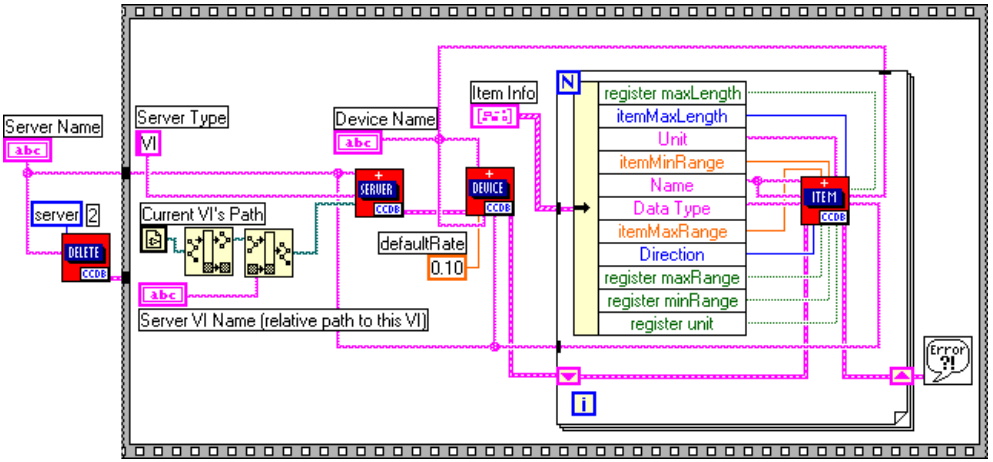


Figure 1-1. Register Dummy Server VI

In Figure 1-1, the VI that registers the server first deletes the existing server registration information from the BridgeVIEW CCDB by calling the SVRG Delete Row VI with the following:

- the **Server Name** to be used (Dummy Server)
- the **Delete What** input set to 2 (server)

This deletes any entry associated with the server name from each of the Server, Device, and Item tables of the CCDB.

Next, the VI supplies new information to the Server, Device, and Item tables by calling the SVRG Add Server Row VI, SVRG Add Device Row VI, and SVRG Add Item Row VI, respectively. See Chapter 2, *Function Reference*, of this document for detailed descriptions of these VIs. In Figure 1-1, the front panel controls of the Register Dummy Server VI store the following as default values:

- **Server Name** (Dummy Server)
- **Server VI Name** (`dmy_srvr.vi`)
- **Device Name** (Dummy Device)
- **Item Info** (item names and parameters)

Although this example server only registers one device, it registers several items for that device.

This example VI, in Figure 1-1, calls the SVRG Add Server Row VI with the following:

- The **server name** to be used (Dummy Server)
- The name of the VI file (`dmy_srvr.vi`) that implements the server
- The path to `dmy_srvr.vi`
- The **server type** set to VI

The path to the configuration utility is left unwired, which indicates there is no configuration utility. The **server type** set to `VI` notifies BridgeVIEW that the server is a VI-based server so BridgeVIEW launches the VI corresponding to that server when the server is selected by a given tag configuration. The Register Dummy Server VI computes the path to the `dmy_srvr.vi` by completing the following steps:

1. Obtain the current VI path (the current VI is Register Dummy Server VI).
2. Remove the VI name.
3. Append `dmy_srvr.vi`.

This is feasible because both the Register Dummy Server VI and `dmy_srvr.vi` are in the same file folder. The server configuration path input is left unwired because the server does not have a configuration utility. Passing in an empty path notifies BridgeVIEW there is not a Configuration Utility available for the server. This example VI leaves all other inputs at their default values.

The example VI, in Figure 1-1, calls the SVRG Add Device Row VI once with the following:

- The unique **device name** (Dummy Device)
- The **device address** (Dummy Device)
- The device **default rate**, set at 0.1 seconds
- The **server name** associated with the device

You must register at least one device for a server if you plan to register any items because all items are associated with a specific device. If the server does not handle any devices, choose a default device name such as ALL. Leave all other inputs at their default values.

The Register Dummy Server VI also calls the SVRG Add Item Row VI for each item registered for the Dummy Server. These items are saved in an array of clusters on the front panel. For each item, this example VI registers the following information:

- A unique **item name**
- An **item data type**
- A direction (input, output, or I/O)

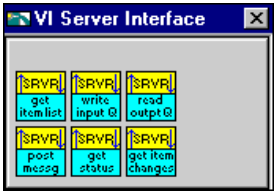
This example VI also registers optional item information, including the following parameters:

- **item max range**
- **item min range**
- **item unit**
- **item max length**

This example VI leaves all other inputs at their default values.

Server Operation

The server uses a set of subVIs to communicate with the BridgeVIEW Engine during server execution. These are contained in the **VI Server Interface** palette shown below.



The servers are launched dynamically when the BridgeVIEW Engine is launched. Servers must execute until their **shutdown** status becomes TRUE. **shutdown** status is returned by several of the server interface VIs.

During server operation, the server uses the VIs listed below to communicate with the BridgeVIEW Engine and to read status from the BridgeVIEW Engine. See Chapter 2, *Function Reference*, of this document for detailed descriptions of these VIs.

- `SRVR get item list.vi`—The SRVR get item list VI returns lists of items, item characteristics, and refnums the BridgeVIEW Engine uses.
- `SRVR write input queue.vi`—The SRVR write input queue VI writes input and I/O item data to the BridgeVIEW Engine. This VI also reports errors on specific input or output items. You can set this VI to return status information regarding whether the server is to shut down or if item changes are pending.
- `SRVR read output queue.vi`—The SRVR read output queue VI receives new output values for output and I/O items from the BridgeVIEW Engine. This VI also returns status information regarding whether the server is to shut down or if item changes are pending.

- SRVR Post Message.vi—The SRVR Post Message VI writes error and non-error messages from the server to the BridgeVIEW Engine where the messages can be logged and displayed to the end-user.
- SRVR get item changes.vi—The SRVR get item changes VI returns a list of item changes which have occurred.

☞ **Note:** ***Because BridgeVIEW 1.0 does not generate server changes while the server is running, you do not need to use the SRVR get item changes VI.***

Server Initialization

When the BridgeVIEW Engine launches a server, the server must first call the SRVR get item list VI, passing in the server name it registered under. This VI returns the list of items the BridgeVIEW Engine uses from the server, as well as details on how to use the listed items. Information specified for each item includes the following parameters:

- **device name**
- **item name**
- BridgeVIEW **datatype**
- **item direction**
- **item datatype**
- **scan rate**
- **notify on change** flag
- BridgeVIEW **refnum** (a signed 32-bit integer)

☞ **Note:** ***The server must use the BridgeVIEW refnums when it passes item information to the BridgeVIEW Engine or receives information from it.***

BridgeVIEW can have multiple tags assigned to an item. The server updates all BridgeVIEW refnums associated with that item. It is best to support this capability. However, if you cannot support this capability, you must send the* can't support multiple connections to item *status for duplicate items in the item list. Refer to the Error Handling and the Status Parameter section of this chapter for more information.

Next, the server sorts through the item list. If any device or item names are incorrect, are not configured for the requested item direction, or cannot be used for some reason, the server writes the status information using the SRVR write input queue VI.

Then, the server polls all valid input or I/O items for their current readings and writes those to the input queue. If there are problems with any items, the appropriate status also must be written to the input queue.

Server Input and Output

The server must run continuously, typically executing two parallel loops, an input polling loop and an output polling loop. Both loops must run until the server is signaled to shut down. In BridgeVIEW 1.0, the BridgeVIEW Engine does not specify the input polling (scan) rates. The user specifies this rate using the Server Configuration Utility. Similarly, the Server Configuration Utility specifies communication parameters, timeouts, and retries.

☞ **Note:** ***BridgeVIEW 1.0 returns* notify on change *set to* TRUE, *and* item datatype *set to default (–1). You can ignore this output for BridgeVIEW 1.0.***

The server polls its inputs according to its polling configuration and writes all new or changed input data to the input queue, along with **timestamp** and **status** information. The SRVR write input queue VI returns the number actually written to the input queue. This notifies the server of any queue overflow situations. Ideally, the queues allocated by the BridgeVIEW Engine are large enough to prevent this situation. By default, the server can instruct BridgeVIEW to block it and handle the rewrite. The server directly handles retries by clearing the **block if queue full** input; however, the server also must check and rewrite data as necessary, otherwise the data is lost.

In addition to polling the item inputs, the server occasionally must read the output queue to obtain item output values.

You wire the **server name** to the SRVR read output queue VI, along with a maximum number of values to read (**max # to read** = 0 reads all available values for the server) and a maximum **timeout** period to wait before reading the queue. The VI returns as soon as one of the following occurs:

- Information is available in the queue
- The server **shutdown** or **changes pending** status is TRUE
- A **timeout** occurs

The server also must write the server input queue using the BridgeVIEW **refnums** corresponding to output items to the input queue, to indicate any **error** status for those items to the BridgeVIEW Engine.

If the item is used as an output, only the value is ignored; however, the status is read from, saved, and then reported. The server must write to the input queue with the status of an output item when that status changes. If a problem occurs when outputting to the item, the server must write to the input queue with the appropriate status. If the status previously was bad but has become good, the server also must write to the input queue with a good status value. Refer to the *Error Handling and the Status Parameter* section of this chapter for more information.

Server Shutdown

When the engine stops, it sends **shutdown** notification to the servers. **shutdown** can be detected from the SRVR write input queue, SRVR read output queue, and SRVR get status VIs. The event-driven SRVR read output queue VI is a good place to wait for shutdown notification because it returns immediately if the engine goes into **shutdown** mode. You can use this mechanism even if the server has no output items. It also might be convenient to explicitly poll the server status occasionally using the SRVR get status VI.

A server is given about 15 seconds to shut down. If the server has not stopped execution by that time, the user is asked for permission to abort (close) the server.

Server Changes

To obtain information about item changes for the server, the server either acquires a completely new **item list** by calling the SRVR get item list VI, or retrieves a list of exceptions only by calling the SRVR get item changes VI. The SRVR get item changes VI specifically lists the items that are obsolete, new, or have changed. Items with no changes associated with them are not included in the outputs from this VI. Calling the SRVR get item list VI or SRVR get item changes VI resets the **changes pending** status.

The server now sorts through the changed item lists. If any device or item names are incorrect, not configured for the requested item direction, or not used for some reason, the server must write the status information using the SRVR write input queue VI.

☞ **Note:** ***BridgeVIEW 1.0 does not change the* item list *while the server is running.***

Sample Server Design

The design of a simple sample server is shown in the following illustration.

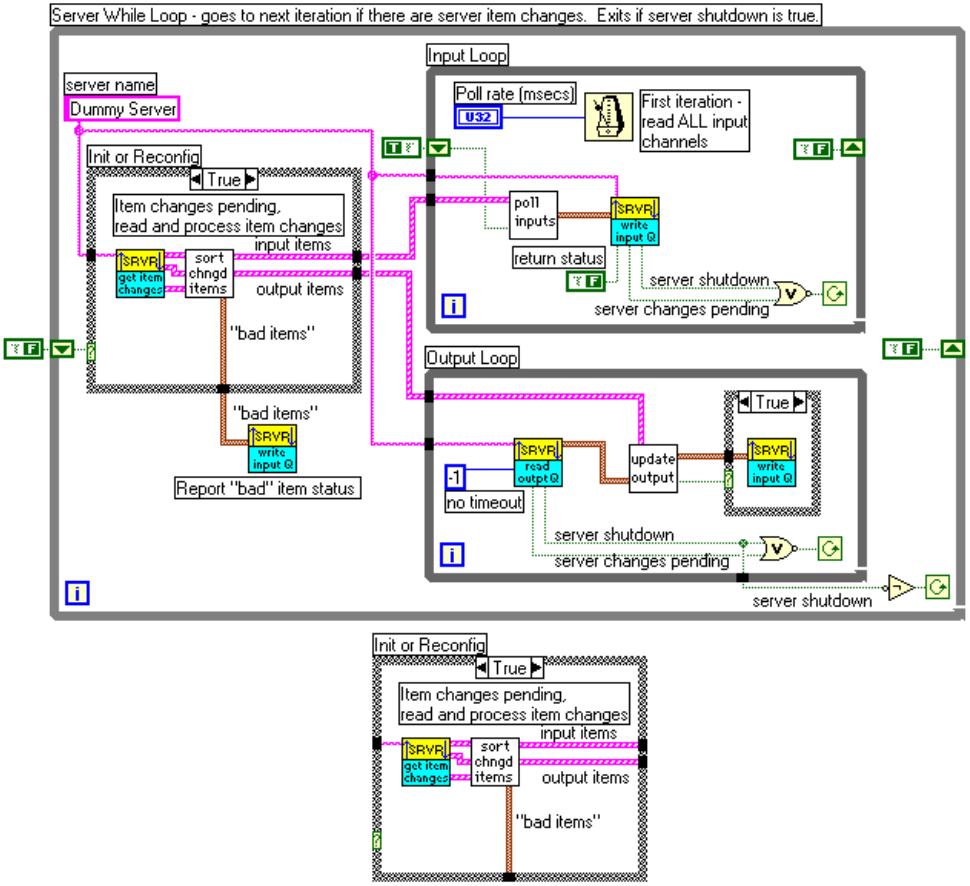


Figure 1-2. Sample VI-Based Server

☞ **Note:** ***This sample server only works with scalar data types, not strings.***

Figure 1-2 shows how the SRVR VIs typically are used. The VIs poll inputs, update outputs, and sort items are server-specific subVIs. The remaining VIs are part of the BridgeVIEW server interface libraries.

The server in Figure 1-2, *Sample VI-Based Server*, is launched for the first time by the BridgeVIEW Engine when a tag configuration using the server is run. First, the server calls the SRVR get item list VI, passing in the **server name** it is registered by (see the *Server Registration* section of this chapter for more information). The server then receives a list of items from the BridgeVIEW Engine to poll. This list includes the following:

- Item strings
- Device strings
- Polling rates
- Need for input, output, or both to be performed for the item
- Data type for the item
- Unique reference number (used by BridgeVIEW to identify the item)

Because BridgeVIEW uses the unique reference number in all subsequent operations, the server must set up internal lookup tables for converting between BridgeVIEW reference numbers and the server representation for each item. You also want to sort out the items that require inputs and those that require outputs, and initialize the server to perform those functions for the requested items.

As part of initially sorting through the **item list**, the server checks errors in the item list. Errors can include the following:

- Unrecognized device
- Unrecognized item
- Unsupported direction
- Wrong data type for item
- Server can't support multiple BridgeVIEW refnums for the item

For any of these conditions, the server must write the appropriate status information to the server input queue for any item or device which is invalid or unusable for any reason. The server does this so BridgeVIEW can mark the bad status for those items in the Real-Time Database.

The server also may post an error message to the BridgeVIEW Engine if the error is considered to be severe, such as not being able to communicate with a device. These messages are displayed to the user.

The server then sets up the following two (or more) loops:

- **Input Loop**—Regularly polls the requested input and I/O items from one or more devices and writes the corresponding value, status, and time stamp information to the server input queue
- **Output Loop**—Waits on any output values for the server to be placed in the server output queue from BridgeVIEW. If any values are read from the output queue, the server writes these values to the output and I/O items

Both the SRVR read output queue and SRVR write input queue VIs return **shutdown** and **changes pending** information for the server. For the SRVR write input queue VI, you must pass in the **server name** and set the **return status** input to TRUE for the status information to return. You also can use the output loop to monitor this condition and not check for the condition in the input loop. In this case, the output loop must notify the input loop to terminate when it detects shutdown. If the **shutdown** status is TRUE, the server then completes execution as soon as possible. If the **changes pending** output is TRUE, the server reads **new item list** or **changed item list** information and adjusts the active items accordingly.

☞ **Note:** ***BridgeVIEW 1.0 does not use the* changes pending *output parameter.***

Error Handling and the Status Parameter

Status is an indication of the quality of the value passed to the server—good, uncertain, or bad. The **status** parameter is stored in the BridgeVIEW Real-Time Database along with the **value** and **timestamp** for each tag. When **status** is less than zero, indicating bad status, the BridgeVIEW Engine assumes that the value for that item is not valid.

If a value is good or uncertain, BridgeVIEW updates the **value**, **timestamp**, and **status** fields in the database with the new information, after scaling the value as necessary. BridgeVIEW also computes alarms and performs historical logging on the value, as was configured for the associated tag.

If a value is bad, BridgeVIEW updates the **timestamp** and **status** fields in the database, but retains the last **value** with a good or uncertain status. BridgeVIEW does not compute alarms levels or log the value in the historical database. A break is recorded instead. Users can activate bad status alarm notification on any tag as part of the tag configuration.

status is a 32-bit signed integer. The top 16-bits (MSW) must be set to one of the status numbers listed in Table 1-1, shown below. The bottom 16-bits (LSW) are used by the server and might be used to pass server specific status information; otherwise leave these bits set at 0. The server determines the appropriate status meaning, and passes the corresponding MSW **status** value. The server specific information is passed to the LSW. The more specific the **status** returned, the better. At the least, the server must indicate whether the **value** is good, uncertain, or bad.

Table 1-1. Status Reports

Quality	MSW Status Value	Status Meaning	Who Reports?
Good	0	No error—Value and timestamp is valid.	Server
Warning—Value Uncertain	50	Is Initial/Default Value.	BridgeVIEW Engine
	60	Value out-of-range. The value is either out of raw-range or out of the engineering unit range during scaling.	BridgeVIEW Engine
	61	Value exceeded high range. The value exceeded the high raw-range or engineering unit range during scaling.	BridgeVIEW Engine
	62	Value exceeded low range. The value exceeded the low raw-range or engineering unit range during scaling.	BridgeVIEW Engine
	100	Uncertain Value.	Server

Table 1-1. Status Reports (Continued)

Quality	MSW Status Value	Status Meaning	Who Reports?
Warning—Value Uncertain (contd.)	105	Last known value (stale data)—Dev comm error. There is a communication error or failure to communicate with the device. This is the last known valid reading for the item. The server must pass a valid value to use this warning status.	Server
	150	Item reading not accurate.	Server
	160	Item value out-of-range.	Server
	161	Item value exceeded high range.	Server
	162	Item value exceeded low range.	Server
Error—Value Bad	–1	BridgeVIEW User Level Error.	BridgeVIEW Engine
	–2	Uninitialized Tag.	BridgeVIEW Engine
	–3	Server Execution Error. The BridgeVIEW engine is unable to find or launch the server.	BridgeVIEW Engine or Server
	–100	Bad Value.	Server
	–101	Unrecognized Device. The server does not recognize the device name string for this item, and cannot acquire or output values.	Server
	–102	Device off-line/out-of-service.	Server
	–103	Device/Item Hardware Error (Hardware Bad). Device and item names are valid, but the server is unable to read or write items because of hardware failure or a configuration error.	Server

Table 1-1. Status Reports (Continued)

Quality	MSW Status Value	Status Meaning	Who Reports?
Error—Value Bad (contd.)	–105	Device Communication Error—failure of communications with device. The device may be temporarily off-line; however, the server is unable to update a value for the item because of lost communication.	Server
	–111	Unrecognized Item. The server does not recognize the item name string for this item, and cannot acquire or output values.	Server
	–112	Unsupported read/write mode. Device and item names are valid, but the server is unable to support the requested read or write mode for the item.	Server
	–113	Unsupported datatype. Device and item names are valid, but the server is unable to support the datatype for the item.	Server
	–114	Unable to support multiple connections to item.	Server

The server must time stamp values even when reporting a bad status.

The server uses the SRVR Post Message VI to post human readable events and errors to the BridgeVIEW Engine system message handler. Use this VI to report catastrophic and general errors, such as losing communication with a device, and the subsequent recovery from such errors. These messages are displayed to the user and logged to a system log file, so be concise and avoid sending excessive messages. As long as things are operating correctly, no messages are necessary. Report these errors once during start-up/initialization and on a per device basis. The server still must pass the appropriate **status** for all requested items on the input queue. If an error message is reported and the server later recovers from the error, the server should send a non-error message notifying the user of the recovery. Remember to be economical when sending these messages. If you constantly send messages, the system

log file for the user fills up and the BridgeVIEW Engine constantly prompts the user.

Performance Issues

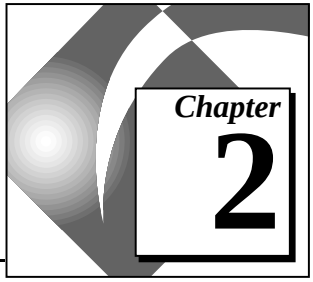
The server must not block BridgeVIEW for long periods of time. If it does, overall operation of the BridgeVIEW Engine and other servers suffers. As long as waiting occurs inside of BridgeVIEW, the BridgeVIEW scheduler multi-tasks all VIs. Blocking can occur if the server makes Code Interface Node (CIN) calls that wait, or if it performs large file writes and reads from BridgeVIEW. Design CIN interfaces so the server can poll inside of BridgeVIEW while it waits for events or timeouts. Design any other type of I/O, such as file writes, to occur in small chunks so little time is taken.

Alternatively, you can use the IA Device Server specification, as documented in the *BridgeVIEW Device Server Toolkit Reference Manual*, to interface DLL-based servers to BridgeVIEW. With this interface, a Server can work with BridgeVIEW as well as other clients, such as LabVIEW®, using the IA Device Server VIs, and CVI™. DLL-based IA Device Servers use multithreading to run parallel with BridgeVIEW.

If most of the server functionality is contained within another application that is interfaced to BridgeVIEW, but effectively runs parallel with BridgeVIEW, you might want to collect input values using a buffer in the server. From BridgeVIEW, you then can poll this queue and pass the information to the BridgeVIEW Engine input queue.

It is possible to send events from an external application to BridgeVIEW through Windows messages. Events then are generated through G occurrences on the block diagram.

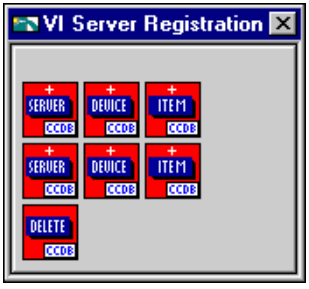
Function Reference



This chapter describes the VIs that register VI-based servers and interface the VI-based servers to the BridgeVIEW Engine during server execution.

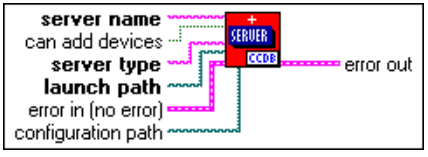
Server Registration VIs

The server uses a set of subVIs to communicate with the BridgeVIEW Engine. These subVIs are contained in the **VI Server Registration** palette shown below.



SVRG Add Server Row VI

The SVRG Add Server Row VI registers your server for use with BridgeVIEW. This VI creates an entry for your server in the Servers table of the CCDB.



server name is the unique name of the server as it should appear in the Server list in the BridgeVIEW Tag Configuration Editor and other utilities. This is the same name you use in your server VI when calling

the SRVR VIs to retrieve information. You must wire a non-empty string for the **server name**, and when possible, use an understandable name. Spaces are permitted in the name, however, the maximum length of the string cannot exceed 255 characters.

can add devices (optional) is set to FALSE, by default. Set this bit to TRUE if your server is capable of interpreting device strings and can accept a new device name at server launch time. When set to TRUE, users can create new device names in the Device list for the server. If your server only can access pre-registered or pre-defined devices, you must set this input to FALSE or leave it unwired.

server type is the type of the server. You must identify your VI-based servers as such by setting the **server type** string to VI. BridgeVIEW then interprets your execution path as a VI path.

launch path is the path to the VI implementing the server, including the VI name. BridgeVIEW locates the path and VI name from the **server name** to dynamically load your server VI.

☞ **Note:** ***The VI name is independent of the* server name.**

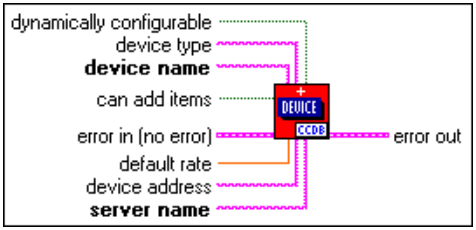
error in (no error) describes the error status before this VI executes.

configuration path (optional) is the path to the configuration utility. If there is no configuration utility, leave this input unwired.

error out describes the error status after this VI executes.

SVRG Add Device Row VI

The SVRG Add Device Row VI registers a device for your server with BridgeVIEW. This VI creates an entry for your device in the Devices table of the CCDB. When you register one or more devices for a server, the **device name** appears in the Devices list when your server is selected in the BridgeVIEW Tag Configuration Editor. Even if you do not have a specific device or all items of interest are associated with a single device, you must register the device if you plan to register any items. In this case, use a default **device name**, such as ALL.



dynamically configurable (optional) is set to TRUE if device values, such as **default rate** can be dynamically reconfigured while running the server. By default, this input is FALSE.

device type (optional) is a string documenting the type of device. This input is for documentation purposes only and might be useful for your server. This field is not used by BridgeVIEW.

device name is the name of the device as it should appear in the Devices list in the BridgeVIEW Tag Configuration Editor and other utilities. Each device registered for your server must have a unique name. Spaces are permitted in the name, however, the maximum length of the string cannot exceed 255 characters.

can add items (optional) is set to FALSE, by default. Set this bit to TRUE if your server is capable of interpreting device strings and can accept a new **device name** at server launch time. When set to TRUE, users can enter new device names in the Devices list for the server. If your server only can access pre-registered or pre-defined devices, you must set this input to FALSE or leave it unwired.

error in (no error) describes the error status before this VI executes.

default rate (optional) is the default sampling period, in seconds, for polling the item.

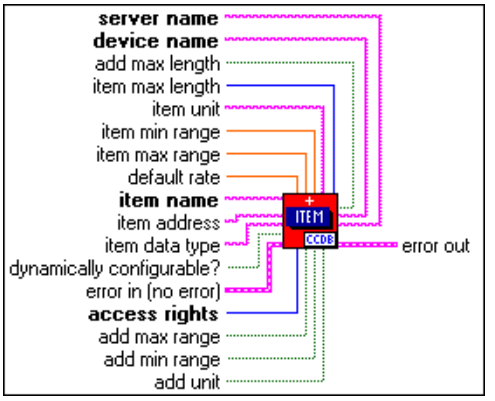
device address (optional) is an input that you can use to record **device address** information stored for this device by your server. BridgeVIEW does not interpret this information.

server name is the name of the server associated with this device. Use the same **server name** you used to register the server. You must wire a non-empty string for **server name**.

error out describes the error status after this VI executes.

SVRG Add Item Row VI

The SVRG Add Item Row VI registers an item for your server device with BridgeVIEW. You must register a device before you can register an item for that device. This VI creates an entry for your item in the Items table of the CCDB.



server name is the name of the server for which this item is registered.

device name is the name of the device for which this item is registered.

add max length is FALSE, by default, and no maximum length value is registered for the item. If you registered a maximum length value, set this input to TRUE.

item max length is the maximum length associated with this item. It is interpreted as the maximum number of bytes in the item for string types (**item data type** = BLOB **or** STR) or the maximum number of bits for bit array types (**item data type** = BITA). If you register the maximum length for the item, you must also set the **add max length** input to TRUE, otherwise the information is not stored in the item row.

With the following three inputs—**item unit**, **item min range**, and **item max range**—you can register engineering unit information for the item. If you register engineering unit information, BridgeVIEW automatically imports the information into the tag configuration when the item is selected. This can be a convenient way of passing configuration information from the server to BridgeVIEW, provided that the server actually has the information. Users still can modify the information. Registering

engineering unit information is optional, and you should only register this information if you can predetermine the actual engineering range and unit information for the item.

item unit (optional) is the engineering unit string for this item. If you register the engineering unit for the item, you must also set the **add unit** input to TRUE, otherwise the information is not stored in the item row.

item min range (optional) is the minimum value in engineering units for this item. If you register the engineering minimum range for the item, you must also set the **add min range** input to TRUE, otherwise the information is not stored in the item row.

item max range (optional) is the maximum value in engineering units for this item. If you register the engineering maximum range for the item, you must also set the **add max range** input to TRUE, otherwise the information is not stored in the item row.

default rate (optional) is the default sampling period, in seconds, for polling the item.

item name is the name of the item as it appears in the Items list in the BridgeVIEW Tag Configuration Editor and other utilities. Each item registered for a device must have a unique name. Spaces are permitted in the name, however, the maximum length of the string cannot exceed 255 characters.

item address (optional) is an input that you can use to record **item address** information stored for this device by your server. This information is not interpreted by BridgeVIEW.

item data type is a string input indicating the item data type, such as Double, Boolean, or Integer. **item data types** are used to predict the type of tag associated with an item when the BridgeVIEW Tag Configuration Editor auto-generates a tag configuration file, as shown in Table 2-1, *Item Data Types*. Users can select any scalar (not STR or BLOB) type for any of the BridgeVIEW tag types: analog, discrete, or bit array. These are all internally represented as a double floating-point value.

Note: ***The ultimate BridgeVIEW datatypes used are*** BLOB ***(*****item data type** = BLOB **or** STR***) or*** DBL ***for all items. BridgeVIEW uses this field to prevent the user from selecting an item with*** **item data type** = BLOB **or** STR ***when configuring an analog, discrete, or bit array tag. BridgeVIEW also uses this field to prevent the user from selecting an item with*** **item data type** ≠ BLOB ***or*** STR ***when configuring a string tag.***

Table 2-1. Item Data Types

Item Data Type String	Actual Data Type	Default BridgeVIEW Tag Type
DBL	G double (8-byte) IEEE float	Analog
BLOB	G string or packed U8 array	String
STR	G string or packed U8 array	String
BOOL	G Boolean	Discrete
I8	G 8-bit signed integer	Analog
I16	G 16-bit signed integer	Analog
I32	G 32-bit signed integer	Analog
U8	G 8-bit unsigned integer	Analog
U16	G 16-bit unsigned integer	Analog
U32	G 32-bit unsigned integer	Analog
SGL	G single (4-byte) float	Analog
BITA	Bit Array up to 32-bit integer.	Bit array

dynamically configurable? (optional) is set to TRUE if item values, such as **default rate** can be dynamically configured when you launch the server. By default, this input is FALSE.

error in (no error) describes the error status before this VI executes.

access rights are the access directions supported by the item. If the item is bi-directional, select I/O. Otherwise, select the input or output direction that is appropriate for an item. BridgeVIEW uses this field to ensure that a tag is configured to access the item in the directions supported by that item. For example, if an item is registered as an input only item, the user only can configure the item as input, when the item is linked to a BridgeVIEW tag.

add max range is FALSE, by default, and no maximum range engineering value is registered for the item. If you registered an **item max range**, set this input to TRUE.

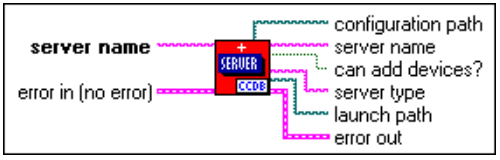
add min range is FALSE, by default, and no minimum range engineering value is registered for the item. If you registered an **item min range**, set this input to TRUE.

add unit is FALSE, by default, and engineering unit is registered for the item. If you registered an **item unit**, set this input to TRUE.

error out describes the error status after this VI executes.

SVRG Get Server Row VI

The SVRG Get Server Row VI returns the information registered for the **server name** from the Server table. You store this information using the SVRG Add Server Row VI.



server name is the registered name of the server.

error in (no error) describes the error status before this VI executes.

configuration path is the path to the configuration utility for the server. An empty path indicates no configuration utility.

server name returns the registered name of the server.

can add devices? returns whether devices can be created for the server.

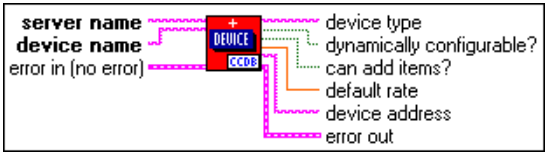
server type returns the type of the server. If it is a VI-based server, the **server type** will be VI.

launch path returns the path of the VI implementing the server, including the VI name.

error out describes the error status after this VI executes.

SVRG Get Device Row VI

The SVRG Get Device Row VI returns the information registered for the **device name** corresponding to **server name**. You store this information using the SVRG Add Device Row VI.



server name is the registered name of the server.

device name is the name of a registered device for the server.

error in (no error) describes the error status before this VI executes.

device type returns a string documenting the type of device. The contents of this string are server specific.

dynamically configurable? returns information on whether device values can be dynamically configured when you launch the server.

can add items? returns information on whether items can be created for the device.

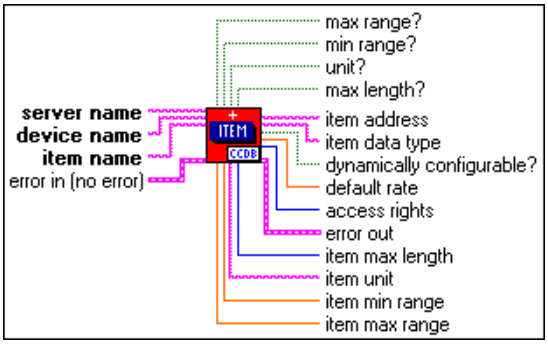
default rate returns the default sampling period, in seconds, for polling the item.

device address returns **device address** information that was stored for this device by your server. The contents of this string are server specific.

error out describes the error status after this VI executes.

SVRG Get Item Row VI

The SVRG Get Item Row VI returns the information registered for **item name** corresponding to the **server name** and **device name**. You store this information using the SVRG Add Item Row VI.



server name is the registered name of the server.

device name is the name of a registered device for the server.

item name is the name of a registered item for the device.

error in (no error) describes the error status before this VI executes.

max range? is TRUE if a maximum range was registered for this item.

min range? is TRUE if a minimum range was registered for this item.

unit? is TRUE if a unit was registered for this item.

max length? is TRUE if a maximum length was registered for this item.

item address returns **item address** information stored for this item. The contents of this string are server specific.

item data type returns the data type stored for this item. Refer to Table 2-1, *Item Data Types*, for more information on the stored data types.

dynamically configurable? returns whether this item can be dynamically configured when you launch the server.

default rate returns the default sampling period, in seconds, for polling the item.

access rights are the access directions supported by the item. I/O indicates the item is bi-directional. Input indicates the item is input only. Output indicates the item is output only.

error out describes the error status after this VI executes.

item max length is the maximum length associated with this item. If the **item data type** = BITA, this number is interpreted to be the number of bits associated with the item. If the **item data type** = STR **or** BLOB, the number is interpreted to be the maximum length the string can be, in bytes. This output is valid only if **max length?** is TRUE.

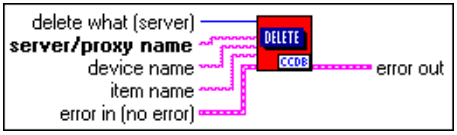
item unit is the engineering unit string for this item. This output is valid only if **unit?** is TRUE.

item min range is the minimum value in engineering units for this item. This output is valid only if **min range?** is TRUE.

item max range is the maximum value in engineering units for this item. This output is valid only if **max range?** is TRUE.

SVRG Delete Row VI

The SVRG Delete Row VI deletes a specific row from the Server, Device, or Item tables. If you delete a server from the Server table, all devices for the server in the Device table, and all items for the server in the Items table are deleted automatically. You do not have to delete devices and items individually if you want to delete them all. Similarly, if a device is deleted from the Devices table, all items for that device in the Items table are deleted automatically.



delete what determines the table from which data is deleted.

2: Delete row from the Server table. This deletes all information associated with the server from the Device and Item tables.

1: Delete row from the Device table. This deletes all information associated with the device from the Device table.

0: Delete row from the Item table. This deletes all information associated with the item from the Item table.

server/proxy name is the name of the server for which the table row is being deleted. You must always enter a server name.

device name is the name of the device for which the table row is being deleted.

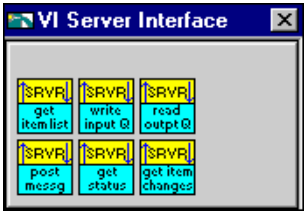
item name is the name of the item for which the table row is being deleted.

error in (no error) describes the error status before this VI executes.

error out describes the error status after this VI executes.

Server Interface VIs

The server also uses a set of subVIs to communicate with the BridgeVIEW Engine during server execution. These VIs are contained in the **VI Server Interface** palette shown below.



SRVR get item list VI

The SRVR get item list VI returns lists of items, item characteristics, and item refnums that the BridgeVIEW Engine requests from a specific server.

server name is the registered name of the server.

item list is the specification for items that the server monitors and controls. **item list** is an array of the `SRVR item list.ctl` Strict Type Definition.

device name is a string containing the name of the device to use. The contents of this string are server specific. For example, you can use the string to pass **device address** information to the server. This string is entered or selected, from a list of preregistered devices, by the user during BridgeVIEW tag configuration. The server must document valid **device name** formats for the user or register a complete list of devices.

item name is a string containing the name of the item to use. The contents of this string are server specific. For example, you may use the string to pass **item address** information and formatting/conversion information to the server. This string is entered, or selected from a list of preregistered items, by the user during BridgeVIEW tag configuration. The server must document valid **item name** formats for the user or register a complete list of items available for each device present.

BVE datatype is a Double or Binary Large Object (DBL or BLOB) data type that the BridgeVIEW Engine requests for the item. Items passed to the BridgeVIEW Engine must be coerced to this datatype.

0: DBL—Indicates you must return a scalar value coerced to a double data type.

1: BLOB—Indicates you must return data as a string or packed U8 array.

item dir can be input, output, or I/O. If **item dir** is input, the item must be polled regularly by the server. The BridgeVIEW Engine cannot control an input item. If **item dir** is output, the item can be controlled only by the BridgeVIEW Engine; that is it will not be monitored. The server does not poll an output item. If **item dir** is I/O, the item must be polled regularly by the server and can be controlled by the BridgeVIEW Engine.

0: input

1: output

2: I/O

item datatype is a data type that the user expects to read from the item. This is normally the default data type for that particular device and item.

–1: use default **item datatype**

0: use DBL **item datatype**

1: use BLOB **item datatype**

Note: ***BridgVIEW 1.0 always passes* –1: use default item datatype.**

scan rate is the rate, in seconds, at which the item is polled by the server. A rate of –1 means you use the default or preconfigured server **scan rate** for this item.

Note: ***BridgeVIEW 1.0 always passes* –1: use default scan rate.**

notify on change If TRUE, pass each new item value read from the device to BridgeVIEW Engine only if it has changed (after server start-up, the server must always return the initial item value). If FALSE, return every item value read from the device, even if it has changed.

Note: ***This parameter is always set to* TRUE *(notify on change only).***

BVE refnum is a BridgeVIEW Engine reference number that must be used by the server when writing an item to the input queue or reading an output item value from the output queue.

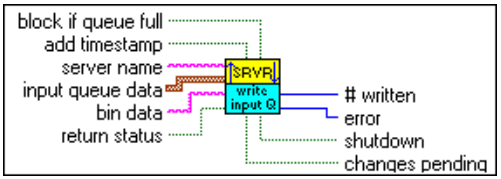
error returns any errors that occurred: 0 = no error; –7202 = unrecognized server name.

shutdown is the notification to end server execution.

Note: ***The same devices and items can occur for multiple BridgeVIEW* refnums. *Servers must handle multiple connections to an item or write the* unable to support multiple connections to item *status to the input queue for duplicate items.***

SRVR write input queue VI

The SRVR write input queue VI writes input item and I/O item data to the BridgeVIEW Engine. This VI also reports item status on specific input, output, or I/O items. You can set this VI to return engine status.

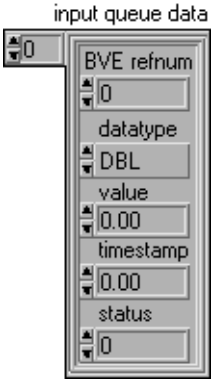


block if queue full By default, this is TRUE. If TRUE, the SRVR write input queue VI blocks if the input queue is full. It continues trying the writes until successful. If FALSE, the server must handle this situation. To handle it, check that the number actually written matches the number of queue entries the server attempted to write. If they are not the same, you must retry with the unwritten entries, or the data will be lost.

add timestamp to input values. Set this input if you have not time stamped the values yourself. By default, this input is FALSE (left unwired), and BridgeVIEW expects the values to have the correct **timestamp** already.

server name (optional) is the registered name of the server. You only need to pass in **server name** if the **return status** input is TRUE. **server name** is used to locate and return server status. Otherwise, **input queue data** is passed to the BridgeVIEW Engine without returning server **shutdown** or **changes pending** status.

input queue data is an array of input item values to pass to the BridgeVIEW Engine. **input queue data** is an array of the `SRVR input queue data.ctl` Strict Type Definition.



BVE refnum is the reference number for an item that was returned by the SRVR get item list VI.

datatype is the Double or Binary Large Object (DBL or BLOB) data type being passed into the BridgeVIEW Engine. **datatype** must correspond to the BridgeVIEW data type specified in the **item list**. BLOB signifies that binary data corresponding to this entry is passed into the **bin data** input.

Note: ***If* status *is bad for the item, the* datatype *parameter is ignored. Therefore, you do not need to set the* datatype *parameter when reporting errors.***

value is the item value when the item value is passed in as a double-precision floating-point number. Scalar values may be interpreted as analog, discrete (Boolean), or bit array (bit vectors up to 32-bits in length), depending on the user tag configuration for a specific device item. All scalar values must be converted to double-floating points to pass to the BridgeVIEW Engine. The server must convert signed or unsigned values to double floating-point numbers correctly. When the item value is passed in as a BLOB, you must put the length in bytes (chars) of the **bin data** string corresponding to this item in the **value** field.

timestamp is a double floating-point number set in seconds since January 1, 1904, (Universal time). Use the **Get Date/Time in Seconds** function to read this time into a VI. The

server can maintain and calculate its own **timestamp** as long as it corresponds to the same seconds since 1904 used by BridgeVIEW.

status is an indication of the quality of the **value** passed to the server-good, uncertain, or bad. See the *Error Handling and the Status Parameter* section of Chapter 1, *VI-Based Server Interface to the BridgeVIEW Engine*, for more information on **status**.

bin data is binary or string data passed to the BridgeVIEW Engine. If any of the input types are set to BLOB, they are passed in the **bin data** input. All BLOBs are treated as strings in BridgeVIEW, and concatenated together in order to create a single string to pass to the BridgeVIEW Engine. For each **input queue data** element writing binary data, the data type in the respective **input queue data** entry is then set to BLOB, and the **value** is set to the length of the string section.

return status If this is set, the SRVR write input queue VI uses the **server name** input to locate and return the server **shutdown** and **changes pending** status.

☞ **Note:** ***You must pass in the* server name *to get a valid indication of* shutdown *or* changes pending *status. If you are not reading these outputs, you do not need to wire the* server name *input.***

 # written indicates the number of **input queue data** entries that were successfully passed to the BridgeVIEW Engine. If this is less than the length of the **input queue data** array, an input queue full condition occurred, and the server writes the remaining data at a later time.

error returns any errors that occurred: 0 = no error; –7201 = write queue full (unable to complete write); –7202 = unrecognized server name.

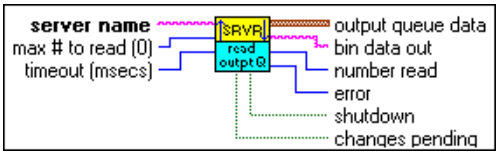
shutdown is the notification to end server execution. **shutdown** is only a valid output if **server name** is passed into the VI. If TRUE, the BridgeVIEW Engine is attempting to stop execution and you must terminate your server execution as soon as possible.

changes pending is only a valid output if **server name** is passed into the VI. If TRUE, changes have been made to you server **item list**. Use the SRVR get item list VI to receive an updated list, or call the SRVR get item changes VI to receive a list of changes to your **item list**. This flag is cleared when the server calls either of these VIs to get the latest **item list** information.

Note: ***BridgeVIEW 1.0 does not use the* changes pending *parameter.***

SRVR read output queue VI

The SRVR read output queue VI receives new output values for output or I/O items from the BridgeVIEW Engine. This VI also returns status information regarding whether the server is to shutdown or if item changes are pending.

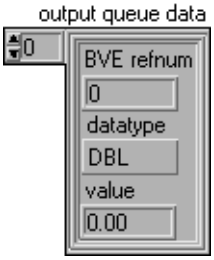


server name is the registered name of the server.

max # to read is the maximum number of values to read from the output queue. If you set **max # to read** = 0, this input reads all available values for the server.

timeout (msecs) is the maximum timeout, in milliseconds, to wait before reading the queue. The VI returns when data is available in the queue for the server or **timeout**, depending on which one occurs first. If the server status changes because of **shutdown** notification or **item list** changes for the server, this VI returns immediately. If **timeout (msec)** = 0, the VI returns immediately. If **timeout (msec)** = –1, the VI waits until data is available in the queue for the server or the server status changes to return. Use a fairly long timeout to prevent unnecessary looping, at least 1 second, or –1 to return only if an event occurs.

output queue data is an array of output item values to be written out to the items by the server. **output queue data** is an array of the SRVR `output queue data.ctl` Strict Type Definition.



BVE refnum is the reference number for an item that was returned by the SRVR get item list VI.

datatype is the Double or Binary Large Object (DBL or BLOB) data type being returned by the BridgeVIEW Engine. **datatype** corresponds to the BridgeVIEW data type specified in the **item list**. BLOB signifies that binary data corresponding to this entry is returned in the **bin data out** output.

value is normally a double floating-point number that the BridgeVIEW Engine wants written to the output item. If the **datatype** is a BLOB, binary data is returned in the **bin data out** output. In this case, **value** signifies the length of the binary string, in the **bin data out** output, corresponding to this entry.

bin data out is binary or string data returned by the BridgeVIEW Engine. If any of the output types are BLOB, they are passed to the **bin data out** output. All BLOBs are treated as strings in BridgeVIEW, and concatenated together into a single string that is returned by this VI. The **datatype** in the respective **output queue data** entry is then set to BLOB, and **value** is set to the length of the string for that entry.

number read indicates the number of **output queue data** entries that have been read from the BridgeVIEW Engine.

error returns any errors that occurred: 0 = no error;
–7202 = unrecognized server name.

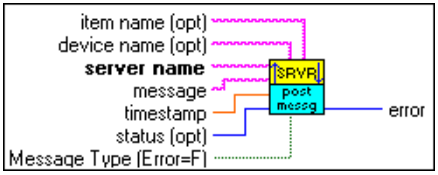
shutdown is the notification to end server execution.

changes pending indicates that changes have been made to your server **item list**. Use the SRVR get item list VI to receive an updated list, or call the SRVR get item changes VI to receive a list of changes to your **item list**. This flag is cleared when the server calls either of these VIs to get the latest **item list** information.

☞ **Note:** ***BridgeVIEW 1.0 does not use the* changes pending *parameter.***

SRVR Post Message VI

The SRVR Post Message VI writes error messages from the server to the BridgeVIEW Engine where the messages can be logged and displayed to the user. See the section *Error Handling and the Status Parameter*, in Chapter 1, *VI-Based Server Interface to the BridgeVIEW Engine*, for more information about when to use this VI.



item name (optional) is the name of the item which caused the error.

device name (optional) is the name of the device which caused the error.

server name is the registered name of the server reporting the error.

message is an ASCII string error message.

timestamp is the timestamp of when the error occurred or is reported.

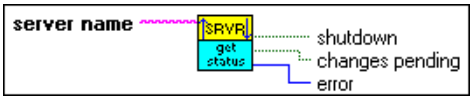
status (optional) is the status associated with the error.

Message Type is the type of message. If FALSE, the message is an error message. If TRUE, the message is not an error, but a server event of interest. By default, **Message Type** is set to FALSE indicating an error message.

error returns any errors that occurred: 0 = no error; –7202 = unrecognized server name.

SRVR get status VI

The SRVR get status VI polls the BridgeVIEW Engine for the current server status.



server name is the registered name of the server.

shutdown is the notification to end server execution.

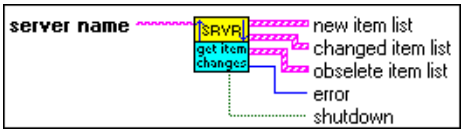
changes pending indicates that changes have been made to your server **item list**. Use the SRVR get item list VI to receive an updated list, or call the SRVR get item changes VI to receive a list of changes to your **item list**. This flag is cleared when the server calls either of these VIs to get the latest **item list** information.

error returns any errors that occurred: 0 = no error; –7202 = unrecognized server name.

Note: ***BridgeVIEW 1.0 does not use the* changes pending *parameter.***

SRVR get item changes VI

The SRVR get item changes VI returns a list of item changes which have occurred.



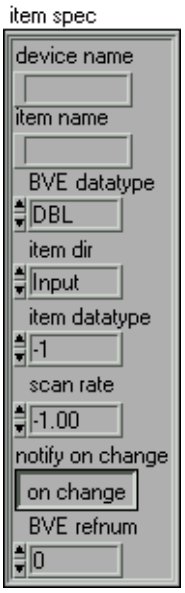
server name is the registered name of the server.

new item list is a list of items that have been added for the server to monitor since the last item update. For parameter definitions, see the *item spec* shown below.

changed item list is a list of items that have been changed in one or more attributes since the last item update. For parameter definitions, see the *item spec* shown below.

obsolete item list is a list of items that the BridgeVIEW Engine no longer wants to monitor. Only the **device name**, **item name**, and **BVE refnum** elements of this list are valid. The server should no longer pass any item values to the server corresponding to these **BVE refnums**. For parameter definitions, see the *item spec* shown below.

The **new item list**, **changed item list**, and **obsolete item list** are all arrays of the `SRVR item list.ctl` Strict Type Definition.



device name is a string containing the name of the device to use. It is updated with any changes made in SRVR get item list VI. The contents of this string are server specific. For example, you can use the string to pass **device address** information to the server. This string is entered or selected, from a list of preregistered devices, by the user during BridgeVIEW tag configuration. The server must document valid **device name** formats for the user or register a complete list of devices.

item name is a string containing the name of the item to use. The contents of this string are server specific. For example, you may use the string to pass **item address** information and formatting/conversion information to the server. This string is entered, or selected from a list of preregistered items, by the user during BridgeVIEW tag configuration.The server must document valid **item name** formats for the user or register a complete list of items available for each device present.

U16

BVE datatype is a Double or Binary Large Object (DBL or BLOB) data type that the BridgeVIEW Engine requests for the item. Items passed to the BridgeVIEW Engine must be coerced to this data type.

0: DBL—Indicates you must return a scalar value coerced to a double data type.

1: BLOB—Indicates you must return data as a string or packed U8 array.

U16

item dir can be input, output, or I/O. If **item dir** is input, the item must be polled regularly by the server. The BridgeVIEW Engine cannot control an input item. If **item dir** is output, the item can be controlled only by the BridgeVIEW Engine; that is it will not be monitored. The server should not poll an output item. If **item dir** is I/O, the item must be polled regularly by the server and can be controlled by the BridgeVIEW Engine.

0: input

1: output

2: I/O

I16

item datatype is a data type that the user expects to read from the item. This is normally the default data type for that particular device and item.

–1: use default **item datatype**

0: use DBL **item datatype**

1: use BLOB **item datatype**

☞ **Note:** ***BridgVIEW 1.0 always passes* –1: use default item datatype.**

DBL

scan rate is the rate, in seconds, at which the item is polled by the server. A rate of –1 means you use the default or preconfigured server **scan rate** for this item.

☞ **Note:** ***BridgeVIEW 1.0 always passes* –1: use default scan rate.**

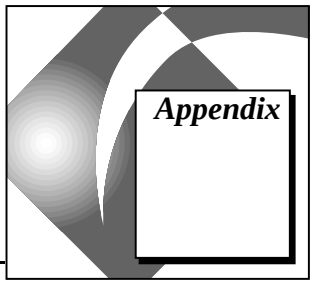
notify on change If TRUE, pass each new item value read from the device to BridgeVIEW Engine only if it has changed (after server start-up, the server must always return the initial item value). If FALSE, return every item value read from the device, even if it has changed.

☞ **Note:** ***This parameter is always set to* TRUE *(notify on change only).***

BVE refnum is a BridgeVIEW Engine reference number that must be used by the server when writing an item to the input queue or reading an output item value from the output queue.

error returns any errors that occurred: 0 = no error; –7202 = unrecognized server name.

shutdown is the notification to end server execution.



Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve your technical problems and a form you can use to comment on the product documentation. When you contact us, we need the information on the Technical Support Form and the configuration form, if your document contains one, about your system configuration to answer your questions as quickly as possible.

National Instruments has technical assistance through electronic, fax, and telephone systems to quickly provide the information you need. Our electronic services include a bulletin board service, an FTP site, a fax-on-demand system, and e-mail support. If you have a hardware or software problem, first try the electronic support systems. If the information available on these systems does not answer your questions, we offer fax and telephone support through our technical support centers, which are staffed by applications engineers.

Electronic Services

Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call (512) 795-6990. You can access these services at:

United States: (512) 794-5422
Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 01 48 65 15 59
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as `anonymous` and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the `/support` directories.

Fax-on-Demand Support

Fax-on-Demand is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access Fax-on-Demand from a touch-tone telephone at (512) 418-1111.

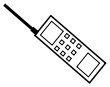
E-Mail Support (currently U.S. only)

You can submit technical support questions to the applications engineering team through e-mail at the Internet address listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

```
support@natinst.com
```

Telephone and Fax Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.

	 Telephone	 Fax
Australia	02 9874 4100	02 9874 4455
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 527 2321	09 502 2930
France	01 48 14 24 24	01 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Israel	03 5734815	03 5734816
Italy	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	5 520 2635	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
U.K.	01635 523545	01635 523154

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___)_____ Phone (___) _____

Computer brand _____ Model _____ Processor_____

Operating system (include version number) _____

Clock speed _____MHz RAM _____MB Display adapter _____

Mouse ___yes ___no Other adapters installed _____

Hard disk capacity _____MB Brand _____

Instruments used _____

_____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is: _____

_____

_____

_____

_____

List any error messages: _____

_____

_____

The following steps reproduce the problem:_____

_____

_____

_____

_____

BridgeVIEW Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

National Instruments Products

DAQ hardware _____

Interrupt level of hardware _____

DMA channels of hardware _____

Base I/O address of hardware _____

Programming choice _____

BridgeVIEW or LabVIEW version _____

Other boards in system _____

Base I/O address of other boards _____

DMA channels of other boards _____

Interrupt level of other boards _____

Other Products

Computer make and model _____

Microprocessor _____

Clock frequency or speed _____

Type of video board installed _____

Operating system version _____

Operating system mode _____

Programming language _____

Programming language version _____

Other boards in system _____

Base I/O address of other boards _____

DMA channels of other boards _____

Interrupt level of other boards _____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: *VI Server Development Toolkit Reference Manual*

Edition Date: March 1997

Part Number: 321297A-01

Please comment on the completeness, clarity, and organization of the document.

_____

_____

_____

_____

_____

_____

_____

If you find errors in the document, please record the page numbers and describe the errors.

_____

_____

_____

_____

_____

_____

_____

Thank you for your help.

Name _____

Title _____

Company _____

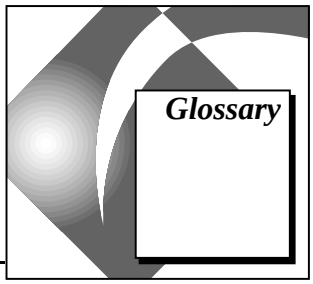
Address _____

_____

Phone (___)_____ Fax (___)_____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
(512) 794-5678



Prefix	Meaning	Value
m-	milli-	10^{-3}
µ-	micro-	10^{-6}
n-	nano-	10^{-9}

Numbers/Symbols

°	degrees
%	percent
Hz	Hertz
sec	seconds

A

A/D	Analog-to-digital.
abort	The procedure that terminates a program when a mistake, malfunction, or error occurs.
address	Character code that identifies a specific location (or series of locations) in memory or on a communications network or device.
alarm	An abnormal process condition. In BridgeVIEW, an alarm occurs if a tag value goes out of its defined alarm limits or if a tag has bad status.

analog tag	A continuous value representation of a connection to a real-world I/O point or memory variable. This type of tag can vary continuously over a range of values within a signal range.
array	An ordered, indexed set of data elements of the same type.
ASCII	American Standard Code for Information Interchange.

B

bit	Binary digit. The smallest possible unit of data: a two-state, yes/no, 0/1 alternative. The building block of binary coding and numbering systems. Several bits make up a *byte*.
bit array tag	A multibit value representation of a connection to a real-world I/O point or memory variable. In BridgeVIEW, this type of tag can be comprised of up to 32 discrete values.
bit vector	A string of related bits in which each bit has a specific meaning.
block diagram	Pictorial description or representation of a program or algorithm. In BridgeVIEW, the block diagram, which consists of executable icons called nodes and wires that carry data between the nodes, is the source code for the virtual instrument. The block diagram resides in the Diagram window of the VI.
Boolean controls and indicators	Front panel objects used to manipulate and display or input and output Boolean (TRUE or FALSE) data. Several styles are available, such as switches, buttons and LEDs.
BridgeVIEW	A G-based (graphical) program development application used commonly for industrial automation purposes.
BridgeVIEW Engine	The heart of the BridgeVIEW (BV) system. It maintains the Real-Time Database of all tag values and alarm states. The BV Engine runs as a separate process, independent of your MMI application.
broken VI	VI that cannot be complied or run; signified by a broken arrow in the Run button.

buffer	Temporary storage for acquired or generated data.
byte	A grouping of adjacent binary digits (bits) operated on by the computer as a single unit.

C

Case structure	Conditional branching control structure, which executes one and only one of its subdiagrams based on its input. It is the combination of the IF THEN ELSE and CASE statements in control flow languages.
channel	Pin or wire lead to which you apply or from which you read the analog or digital signal.
cluster	A set of ordered, unindexed data elements of any data type including numeric, Boolean, string, array, or cluster. The elements must be all controls or all indicators.
Code Interface Node (CIN)	Special block diagram node through which you can link conventional, text-based code to a VI.
command	A directive to a device.
CCDB	Common Configuration Database. Manages the registered BridgeVIEW Server information by maintaining tables of servers, devices, and items.
connector	Part of the VI or function node that contains its input and output terminals, through which data passes to and from the node.
control	Front panel object for entering data to a VI interactively or to a subVI programmatically.
CPU	Central processing unit.

D

data acquisition	Process of acquiring data, typically from A/D or digital input plug-in boards.
datatype descriptor	Code that identifies datatypes, used in data storage and representation.

DDE	Dynamic Data Exchange. A client-controlled Windows protocol for communication between applications.
device	An instrument or controller that is addressable as a single entity and controls or monitors real-world I/O points. A device is often connected to the host computer through some type of communication network, or can be a plug-in device.
device server	An application that communicates with and manages a peripheral hardware device such as a Programmable Logic Control (PLC), remote I/O device or plug-in device. Device servers pass item values to the BridgeVIEW Engine in real time.
diagram window	A VI window that contains the VI's block diagram code.
dialog box	An interactive screen with prompts in which the user specifies additional information needed to complete a command.
discrete tag	A two-state (on/off) value representation of a connection to a real-world I/O point. In BridgeVIEW, this type of tag can be either a one (TRUE) or a zero (FALSE).
DLL	Dynamic link library.

E

Engine	*See* BridgeVIEW Engine.
engineering units (EU)	Terms of data measurement, as degrees Celsius, pounds, grams and so on.
error message	An indication of a software or hardware malfunction, or an unacceptable data entry attempt.
event	Something that happens to a tag in the BridgeVIEW system. Events include tags going into or out of alarm state and the user setting a tag value.
event driven programming	A method of programming whereby the program waits on an event occurring before executing one or more functions.
executable	A stand-alone piece of code that will run, or execute.

F

FIFO	First-In-First-Out; a method of data storage in which the first element stored is the first one retrieved.
For Loop	Iterative loop structure that executes its subdiagram a set number of times. Equivalent to conventional code: `For i = 0 to n - 1, do ....`
front panel	The interactive user interface of a VI. Modeled from the front panel of physical instruments, it is composed of switches, slides, meters, graphs, charts, gauges, LEDs, and other controls and indicators.
function	Built-in execution element, comparable to an operator, function, or statement in a conventional language.

G

G	Graphical programming language used to develop BridgeVIEW applications.

H

Hz	Hertz. The number of scans read or updates written per second.

I

I/O	Input/output. The transfer of data to or from a computer system involving communications channels, operator interface devices, and/or data acquisition and control interfaces.
icon	Graphical representation of a node on a block diagram.
IEEE	Institute of Electrical and Electronic Engineers.
indicator	Front panel object that displays output.
Input/Output (I/O) tag	A tag that accepts Real-Time Database values from a device server and sends values to the server.

item	A channel or variable in a real-world device that is monitored or controlled by a BridgeVIEW device server.
iteration terminal	The terminal of a For Loop or While Loop that contains the current number of completed iterations.

L

LabVIEW	Laboratory Virtual Instrument Engineering Workbench. A G-based (graphical) program development application used commonly for test and measurement purposes.
LED	Light-emitting diode.
LSW	Least Significant Word.

M

Man Machine Interface (MMI)	A graphical user interface for the user to interact with the BridgeVIEW system.
MB	Megabytes of memory.
MSW	Most Significant Word.
multitasking	The ability of a computer to perform two or more functions simultaneously without interference from one another. In operating system terms, it is the ability of the operating system to execute multiple applications/processes by time-sharing the available CPU resources.

N

nodes	Execution elements of a block diagram consisting of functions, structures, and subVIs.

O

object	Generic term for any item on the front panel or block diagram, including controls, nodes, wires, and imported pictures.
OLE	Object Linking and Embedding.
OLE Automation	A feature which allows BridgeVIEW to access objects by automation servers in the system.
operator	The person who initiates and monitors the operation of a process.

P

palette	A display of pictures that represent possible options.
Panel window	VI window that contains the front panel, the execution palette, and the icon/connector pane.
path	Description of the location of a file or directory, including the volume containing the file or directory, the directories between the top level and the file or directory, and the file or directory name.
polling	A method of sequentially observing each I/O point or user interface control to determine if it is ready to receive data or request computer action.
Programmable Logic Control (PLC)	A device with multiple inputs and outputs that contains a program you can alter. BridgeVIEW Device Servers establish communication with PLCs.

Q

query	Like a *command*, causes a device to take some action, but requires a response containing data or other information. A command does not require a response.
queue	A group of items waiting to be acted upon by the computer. The arrangement of the items determines their processing priority. Queues usually are accessed in a FIFO fashion.

R

RAM	Random access memory.
range	The region between the limits within which a quantity is measured, received, or transmitted expressed by stating the lower and upper range values.
read	To get information from any input device or file storage media.
Real-Time Database (RTDB)	An in-memory snapshot of all tags in the system.
refnum	An identifier of a DDE conversation or open files that can be referenced by related VIs.
register	A high-speed device used in a CPU for temporary storage of small amounts of data or intermediate results during processing.

S

sampling period	The time interval between observations in a periodic sampling control system.
scalar	Number capable of being represented by a point on a scale. A single value as opposed to an array. Scalar Booleans, strings and clusters are explicitly singular instances of their respective data types.
scan rate	The number of times (or scans) per second that a device acquires data from channels. For example, at a scan rate of 10Hz, a device samples each channel in a group 10 times per second.
server	The application that receives messages and requests from the client application.
Sequence structure	Program control structure that executes its subdiagrams in numeric order. Commonly used to force nodes that are not data dependent to execute in a desired order.
signed integer	*n* bit pattern, interpreted such that the range is from $-2(n-1)$ to $+2(n-1)-1$.
string	A connected sequence of characters or bits treated as a single data item.

string tag	A string representation of a connection to a real-world I/O point.
structure	Program control element, such as a Sequence, Case, For Loop, or While Loop.
subdiagram	Block diagram within the border of a structure.
subVI	A VI used in the block diagram of another VI.
system errors	Errors that happen in the BridgeVIEW system, like a server going down. System errors are displayed in a dialog box, on the Engine User Interface, and also are logged in a syslog file.
system events	Events that occur in the BridgeVIEW system, like an operator logging on or a utility starting up. System events are logged in a syslog file.

T

tag	A connection to a real-world I/O point or a memory variable. Tags can be one of four datatypes: analog, binary, discrete, or string.
tag attributes	Parameters pertaining to a tag, like its alarm, limits, or Engineering Units. Tag attributes are configured in the Tag Configuration Editor but can be changed dynamically using the Tag Attributes VIs.
Tag Browser	A utility to view the configuration parameters of a tag, as configured in the Tag Configuration Editor.
Tag Configuration Editor	A utility to configure various parameters of a tag, such as connection information, scaling, or logging.
Tag Monitor	A utility to view the current value of a tag, along with its status and alarm state.
tag status	A variable that determines the validity of a tag value. A negative status represents an error, a positive status represents a warning, and a status of zero represents a good tag value.
timeout	The time (in milliseconds) that a VI waits for an operation to complete. Generally, a timeout of –1 causes a VI to wait indefinitely.

timestamp — The exact time and date at which a tag value was sampled. Tag values are stored with their timestamps in the RTDB.

toolbar — Bar containing command buttons that you can use to run and debug VIs.

U

unsigned integer — *n* bit pattern interpreted such that the range is from 0 to $2n - 1$.

user — *See* operator.

utility — A program that helps the user run, enhance, create, or analyze other programs and systems.

V

VI Library — Special file that contains a collection of related VIs for a specific use.

virtual instrument (VI) — A program in the graphical programming language G; so-called because it models the appearance and function of a physical instrument.

W

While Loop — Post-iterative test loop structure that repeats a section of code until a condition is met. Comparable to a Do loop or a Repeat-Until loop in conventional programming languages.

wire — Data path between nodes.