LabVIEW™

VI-Based Server Development Toolkit Reference Manual

March 2003 Edition
Part Number 370260B-01

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Worldwide Offices

Australia 61 2 9672 8846, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599,
Canada (Calgary) 403 274 9391, Canada (Montreal) 514 288 5722, Canada (Ottawa) 613 233 5949,
Canada (Québec) 514 694 8521, Canada (Toronto) 905 785 0085, Canada (Vancouver) 514 685 7530,
China 86 21 6555 7838, Czech Republic 420 2 2423 5774, Denmark 45 45 76 26 00,
Finland 385 0 9 725 725 11, France 33 0 1 48 14 24 24, Germany 49 0 89 741 31 30, Greece 30 2 10 42 96 427,
Hong Kong 2645 3186, India 91 80 51190000, Israel 972 0 3 6393737, Italy 39 02 413091,
Japan 81 3 5472 2970, Korea 82 02 3451 3400, Malaysia 603 9059 6711, Mexico 001 800 010 0793,
Netherlands 31 0 348 433 466, New Zealand 64 09 914 0488, Norway 47 0 32 27 73 00,
Poland 48 0 22 3390 150, Portugal 351 210 311 210, Russia 7 095 238 7139, Singapore 65 6 226 5886,
Slovenia 386 3 425 4200, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 0 8 587 895 00,
Switzerland 41 56 200 51 51, Taiwan 886 2 2528 7227, United Kingdom 44 0 1635 523545

For further support information, refer to the *Technical Support and Professional Services* appendix. To comment on the documentation, send email to techpubs@ni.com.

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 document 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

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or `ni.com/patents`.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Conventions

The following conventions are used in this manual:

»	The **»** symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
	This icon denotes a note, which alerts you to important information.
bold	Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.
italic	Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.
`monospace`	Text in this font denotes text or characters that you should enter 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, operations, variables, filenames and extensions, and code excerpts.

Contents

1 Introduction

This document gives an overview of VI-based servers, describes the development tools, and explains the steps for developing and testing VI-based servers. For information about the basic functionality of the DSC Module, refer to the *LabVIEW Datalogging and Supervisory Control Module Developer Manual*.

VI-based servers are not the same as a VI Server. The *VI Server* is used to programmatically control front panel objects, VIs, and LabVIEW and to dynamically load, edit, and run VIs on the computer or remotely across a network. A *VI-based server* is a data server that is written in LabVIEW, unlike an OPC server, which is typically written in Visual Basic or C++. Like any server, a VI-based server can be a real I/O server for a device, a simulation server as in the Cookie Factory example, or a computational server doing complex computation for your application and serving the results as items.

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

Note Three cases exist in which you do not use LabVIEW to interface with the Tag Engine. If a server is written as a DLL to the National Instruments Industrial Automation Device Server Specification (IA Device Server), the Tag Engine can interface to it through the IAIO Server Proxy. Also, if a server is implemented as a Windows DDE server or an OPC server, the Tag Engine can interface to it directly.

VI-Based Server Development Tools

The LabVIEW Datalogging and Supervisory Control (DSC) Module includes the following tools for developing and testing VI-based servers.

- When you install the DSC Module, the **DSC Server Development** palette is added to the **Functions** palette. This palette contains the Server Interface VIs and Server Registration VIs. A **DSC Server Data**

Types palette also is added to the **Controls** palette. Refer to the *LabVIEW Help* for detailed descriptions of the DSC Server Development VIs.

- You can find sample servers, sample server registration VIs, and sample server configuration utility VIs in the `LabVIEW\examples\lvdsc\Servers\` directory to help you get started.
- You can use the Server Browser utility to interactively view the devices, items, and capabilities registered by a server. With this utility, you can launch a server configuration utility (if available) or, from the Tag Engine, display the front panel of a server while it is running. Select **Tools»DSC Module»Advanced»Server Browser**.
- Use the Interactive Server Tester to emulate the Tag Engine/Server Interface in LabVIEW. With this tool, you can test launching, error reporting, and reading and writing server data in a full G development system environment. Select **Tools»DSC Module»Advanced» Interactive Server Tester**.
- For more information about these tools, open the Context Help window (<Ctrl-H>) when you use them.

Overview of the Server Development Process

The server development process involves several steps, discussed in the following sections.

Create the Server

Build a server using the Server Interface VIs. Refer to the *Server Operation* section of Chapter 2, *VI-Based Server Interface to the Tag Engine*, for information regarding the architecture of a server and which VIs to use. This toolkit includes a server named `dummy server.vi` in the `examples\lvdsc\Servers\Sample` folder. In addition to this sample server, several of the VI-based servers shipped with the DSC Module, such as the SIM Server, are in source code form.

Register the Server

In order for the DSC Module to launch a VI-based server and allow a user to configure tags that use the server, you must register the server. Refer to the *Server Registration* section of Chapter 2, *VI-Based Server Interface to the Tag Engine*, for information on the VIs to use to register a server. This toolkit includes a server registration example named `Register Dummy Server.vi` in the `examples\lvdsc\Servers\Sample` folder.

Debug the Server

This toolkit provides a utility called the Interactive Server Tester utility with which you load and run a server in LabVIEW, allowing you to debug the server while it is running. Refer to the *Debugging and Testing Your Server* section of Chapter 2, *VI-Based Server Interface to the Tag Engine*, for more information.

View the Server Running in the Tag Engine (Optional)

When the server is running in the Tag Engine, you cannot debug it or view its block diagram, but you can view the server front panel.

Click the **View Servers in Use** button on the Engine Manager front panel to launch the Server Browser utility. Select the server in the server list. A diamond symbol appears next to the server name if the server is loaded, and the diamond is black if the server is running. Click the **Show Server User Interface** button to open the server and show the front panel while it is running. By adding indicators to the server front panel, you can monitor server operation while it is running in the Tag Engine.

Create a Server Configuration Utility (Optional)

Depending on the complexity of the server, you might choose to provide a server configuration utility rather than just a server registration VI. Relevant server configuration information can be stored in a file and retrieved by the server at run time. This configuration utility then also registers the server information. This toolkit includes a server configuration example called Server Configuration Template in the `Srvr_cfg.llb`, available in the `examples\lvdsc\Servers\Sample` folder.

Deploying VI-Based Servers

You need to follow certain steps if you want to distribute a VI-based server. For more information about deploying a VI-based server, refer to `ni.com/zone`.

2

VI-Based Server Interface to the Tag Engine

This chapter describes VI-based server configuration and registration, server operation, and error handling and performance issues you might encounter.

Data Types and Timestamps

VI-based servers supply data points from several input items to the Tag Engine as these points are read. The Tag Engine also can send values for output items. By default, the DSC Module ignores server timestamps. To set the DSC Module to accept server timestamps for a VI-based device server you create, set the `UseServerTimestamps=false` setting to `true` in the `dscengine.ini` file in the `LabVIEW` directory.

The Tag Engine accepts numeric (double precision) values and string data from servers. Double values can be analog, discrete (Boolean), or bit array (bit vectors up to 32 bits in length), depending on the tag configuration for a specific device item. String data is a packed array of unsigned 8-bit integers. All scalars must be converted to double precision floating-point values to pass to the Tag Engine. The server converts correctly signed or unsigned values to double floating-point representations.

Ideally, the server timestamps 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 precision floating-point number rather than an unsigned 32-bit integer. Therefore, resolution is less than 1 second. If the server cannot timestamp the values as they are acquired, the server can instruct the Tag Engine to timestamp the values when they are received.

Server Configuration

Generally, a server has a configuration utility associated with it that allows you to complete the following tasks:

- Set up communication parameters
- Specify error handling
- Configure hardware
- Configure poll rates
- Define a set of valid device and item names

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

During configuration, the server must register information about itself and the devices and items it manages. Although servers are not required to have configuration utilities, they must be registered before the LabVIEW Datalogging and Supervisory Control (DSC) Module can use them.

Refer to the Server Configuration Template VI in the `examples\lvdsc\Servers\Sample\Srvr_cfg.llb` for an example server configuration utility.

Server Registration

The DSC Module uses the Common Configuration Database (CCDB) file to locate present servers and retrieve details about those servers, such as paths, registered devices, and registered items.

This database maintains tables of servers, devices, and items. Register information about your server as part of your server configuration utility. If you lack a configuration utility for your server, you must provide a VI that performs the registration.

The server uses a set of subVIs to configure data in the configuration database. These subVIs are contained in the **DSC Module Server Registration** palette shown in Figure 2-1.

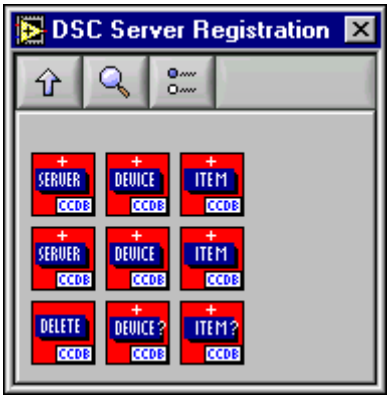

Figure 2-1. DSC Module Server Registration Palette

To register your server, use the SVRG Add Server Row VI, which 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 Tag Configuration Editor server list with the name supplied in **server name**. To use your server from the DSC Module, you must register the following information:

- **server name**—name used by the server in the block diagram when using the DSC Server Interface VIs
- **launch path** to 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** (refer to Table 2-1)
  - allowed item directions (**access rights**): input, output, I/O (required)
  - **item min range** and **item max range** (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. This VI creates an entry for your device in the Devices table of the CCDB. You are not required to

register a device if your server can interpret device strings to identify 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 you select that server in the 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. 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. You are not required to register an item if your server can interpret item strings to identify the device 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 Tag Configuration Editor.

Registering engineering unit information is optional and should be done only 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 when creating tags in the Tag Configuration Editor.

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 these VIs to verify whether your information is registered successfully.

The **item data type** parameter is a string input indicating the item data type, such as Double, Boolean, or Integer. This parameter is used to predict the type of tag associated with an item when the Tag Configuration Editor auto-generates a tag configuration file, as shown in the following table. Users can select any scalar type (not STR or BLOB) for any of the tag types—analog, discrete, or bit array—that are internally represented as double floating-point values.

Table 2-1. Item Data Types

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

Note The ultimate data types used are BLOB (**item data type** = BLOB or STR) or DBL for all items. LabVIEW 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. LabVIEW also uses this field to prevent the user from selecting an item with item data type ≠ BLOB or STR when configuring a string tag. The DSC Module treats items with **item max length** greater than 1, excluding BITA, as string tags.

Register Server Example

The Register Dummy Server VI, shown in Figures 2-2 and 2-3, illustrates how to register information for your server. The user can configure server behavior, devices, or communication 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. You might not develop a server configuration utility in other cases, such as a simple device or fixed server configuration or if you are writing a server to simulate tags. In these cases, you must develop a VI similar to the Register Dummy Server VI in Figures 2-2 and 2-3 and register the items for which your server generates or accepts data. The `examples\lvdsc\Servers\Sample` folder includes this VI and a more complete server configuration utility VI example. Most of the simulation server examples for the DSC Module have a register server VI similar to the Register Dummy Server VI.

In the Register Dummy Server VI, the VI that registers the server first deletes the existing server registration information from the CCDB by calling the SVRG Delete Row VI with the following information:

- **server/proxy name** (`Dummy Server`)
- **delete what** input set to `2` (Server table)

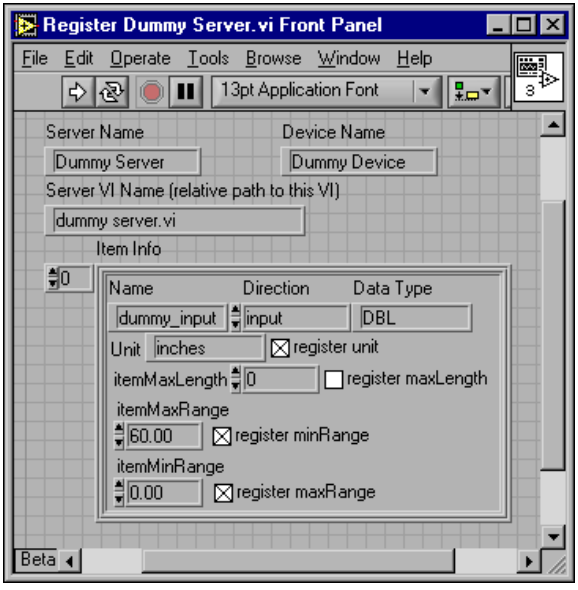

Figure 2-2. Register Dummy Server VI Front Panel

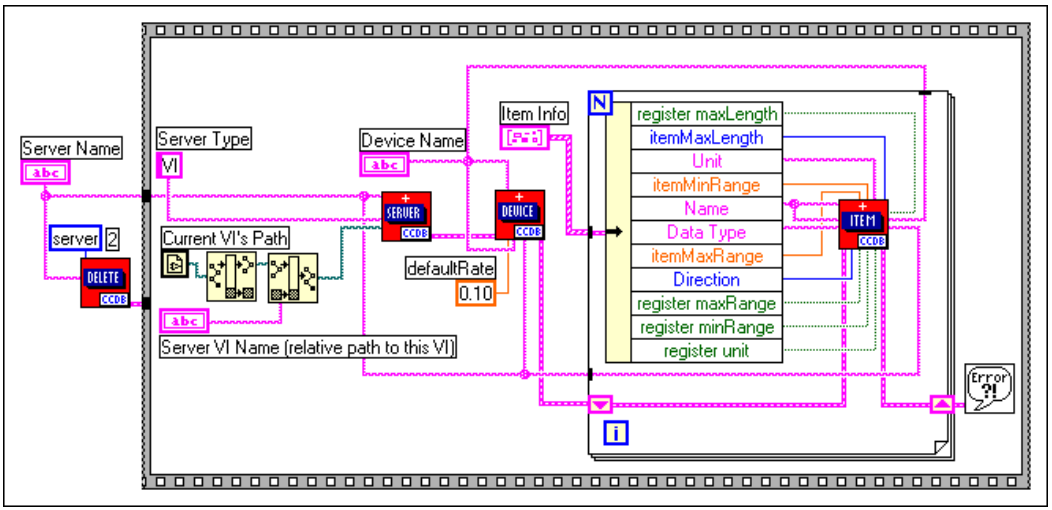

Figure 2-3. Register Dummy Server VI Block Diagram

Running this VI 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. Refer to the *LabVIEW Help* for detailed descriptions of these VIs.

The front panel controls of the Register Dummy Server VI store the following as default values:

- **Server Name** (`Dummy Server`)
- **Server VI Name** (`dummy server.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.

The Register Dummy Server VI calls the SVRG Add Server Row VI with the following information:

- server name (`Dummy Server`)
- name of the VI file (`dummy server.vi`) that implements the server
- path to `dummy server.vi`
- server type set to VI

The path to the configuration utility is left unwired to indicate that no configuration utility exists. Set **server type** to `VI` to notify the DSC Module that the server is a VI-based server. The DSC Module 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 `dummy server.vi` by obtaining the current VI path (current VI is Register Dummy Server VI), removing the VI name, and appending `dummy server.vi`.

The VI can compute the path because both the Register Dummy Server VI and `dummy server.vi` are in the same file folder. This example VI leaves all other inputs at their default values.

The Register Dummy Server VI calls the SVRG Add Device Row VI once with the following information:

- unique **device name** (`Dummy Device`)
- **device address** (`Dummy Device`)
- **default rate**, set at `0.1` seconds
- **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:

- unique **item name**
- **item data type**
- 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 Tag Engine during server execution. These are contained in the **DSC Module Server Interface** palette shown in Figure 2-4.

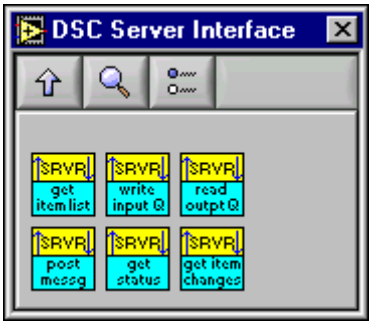

Figure 2-4. DSC Module Server Interface Palette

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

During operation, the server uses the following VIs to communicate with the Tag Engine and to read the status from the Tag Engine. Refer to the *LabVIEW Help* for detailed descriptions of these VIs.

- SRVR Get Item List VI—Returns lists of groups and items, item characteristics, and refnums the Tag Engine uses.
- SRVR Write Input Queue VI—Writes input and I/O item data to the Tag Engine. This VI also reports errors on specific input or output items. You can set this VI to return status information specifying whether the server is to shut down or whether item changes are pending.

- SRVR Read Output Queue VI—Receives new output values for output and I/O items from the Tag Engine. This VI also returns status information specifying whether the server is to shut down or whether item changes are pending.
- SRVR Post Message VI—Writes error and non-error messages from the server to the Tag Engine where the messages can be logged and displayed to the user.
- SRVR Get Status VI—Polls the Tag Engine for the current server status.
- SRVR Get Item Changes VI—Returns a list of group and item changes.

Server Initialization

When the Tag Engine launches a server, the server first must call the SRVR Get Item List VI, passing in the server name it registered under. This VI returns the list of items the Tag Engine uses from the server and details on how to use the listed items. This VI also returns a list of groups that specify timing information for the items. Information specified for each item includes the following parameters:

- device name
- item name
- BVE datatype
- item direction
- item datatype
- scan rate
- notify on change flag
- BVE refnum (a signed 32-bit integer)
- group name
- access path

Information specified for each group includes the following parameters:

- group name
- scan rate
- % deadband
- device name

Note The server must use the refnums when it passes item information to the Tag Engine or receives information from it.

You can have multiple tags assigned to an item. The server updates all refnums associated with that item. Although it is best to support this capability, if you cannot, 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, not configured for the requested item direction, or not available for some reason, the server writes the status information using the SRVR Write Input Queue VI. The server can use the group list to determine the timing configuration for each item. The **scan rate** and **device name** information is already duplicated in the item list. One additional parameter, **% deadband**, is available only from the group list. A server can ignore the group list if it does not implement **% deadband**.

Finally, 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 is written to the input queue.

Server Input and Output

The server must run continuously, usually 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. The server should configure timing for input polling to match the **scan rate** requested in the item list or as close to the specified **scan rate** as possible.

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, which notifies the server of any queue overflow situations. Ideally, the queues allocated by the Tag Engine are large enough to prevent overflow. By default, the server can instruct the DSC Module to block the server 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, or the data is lost.

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

Wire the **server name** input 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 when one of the following events occurs:

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

To update the status of a data item, the server must send a data update to the Tag Engine by writing a new value for the item to the server input queue. This applies to output-only items as well as input-only and input/output items. When you update an output-only data item in this manner, the data value in the update is ignored, but the status of the item is changed.

When the status of a data item changes, the server must update the status of the item. Refer to the *Error Handling and the Status Parameter* section of this chapter for more information.

Server Shutdown

When the Tag Engine stops, it sends shutdown notification to the servers. Shutdown can be detected from the SRVR Write Input Queue VI, SRVR Read Output Queue VI, and SRVR Get Status VI. The event-driven SRVR Read Output Queue VI is a good place to wait for shutdown notification because it returns immediately if the Tag Engine goes into shutdown mode. You can use this mechanism even if the server has no output items. You also can explicitly poll the server status occasionally using the SRVR Get Status VI.

A server is given about 30 seconds to shut down by default. 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 and group list by calling the SRVR Get Item List VI or retrieves lists of exceptions by calling the SRVR Get Item Changes VI. The SRVR Get Item Changes VI specifically lists the items and groups that are obsolete, new, or have changed. Items with no associated changes are not included in the outputs. 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 and group 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.

Sample Server Design

Figure 2-5 shows the design of a simple server.

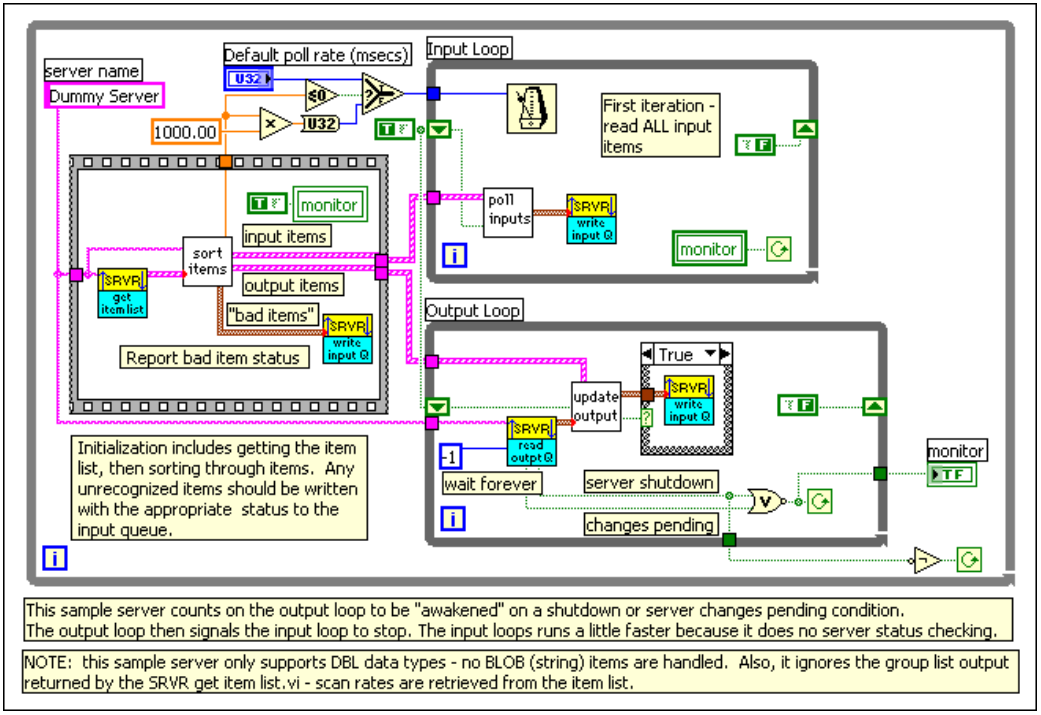

Figure 2-5. Sample VI-Based Server

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

Figure 2-5 demonstrates how the SRVR VIs are often used. The VIs that poll inputs, update outputs, and sort items are server-specific subVIs. The remaining VIs are part of the DSC Module Server Interface libraries.

The server in Figure 2-5 is launched for the first time by the Tag 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** under which it is registered. Refer to the *Server Registration* section of this chapter for more

information. The server then receives a list of items from the Tag Engine to poll, including the following elements:

- 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 the DSC Module uses to identify the item

This server only uses the item information and ignores the group output SRVR Get Item List VI.

Because the DSC Module uses the unique reference number in all subsequent operations, you must configure the server to set up internal lookup tables for converting reference numbers and the server representation for each item. Sort out the items that require inputs and those that require outputs, and initialize the server to perform those functions for the requested items.

While initially sorting through the item list, the server checks for errors in the item list, including the following errors:

- Unrecognized device
- Unrecognized item
- Unsupported direction
- Wrong data type for item
- Server cannot support multiple refnums for the item

For any of these conditions, the server writes the appropriate status information to the server input queue for any item or device that is invalid or unusable for any reason, and the DSC Module marks the bad status for those items.

If the error is considered severe, such as not being able to communicate with a device, the server might post an error message to the Tag Engine. 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 timestamp 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 the DSC Module. 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 VI and SRVR Write Input Queue VI return **shutdown** and **changes pending** information for the server. For the SRVR Write Input Queue VI, pass in the **server name** and set the **return status** input to TRUE for the status information to return. If you use the output loop to monitor this condition, the input loop does not need to check for the condition. 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.

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 along with the value and timestamp for each tag. When **status** is less than zero, indicating bad status, the Tag Engine assumes that the value for that item is not valid.

If a value is good or uncertain, LabVIEW updates the value, timestamp, and status with the new information, after scaling the value as necessary. The DSC Module also computes alarms and performs historical logging on the value, as was configured for the associated tag.

If a value is bad, LabVIEW updates the timestamp and status but retains the last value with a good or uncertain status. LabVIEW does not compute alarm levels. 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 2-2. The bottom 16 bits (LSW) are used by the server, sometimes 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; however, the server must indicate if the **value** is good, uncertain, or bad.

Table 2-2. Status Reports

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

Table 2-2. Status Reports (Continued)

Quality	MSW Status Value	Status Meaning	Reporter
Error—Value Bad	–1	User level error.	Tag Engine
	–2	Uninitialized tag.	Tag Engine
	–3	Server execution error. The Tag Engine is unable to find or launch the server.	Tag 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 offline/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
	–105	Device communication error—Failure of communications with device. The device might be temporarily offline; 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 data type. 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 timestamp values even when reporting a bad status.

The server uses the SRVR Post Message VI to post human-readable events and errors to the Tag 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.

Note Remember to be economical when sending messages. If you send messages frequently, the system log file for the user fills up and the Tag Engine constantly prompts the user.

Server-User Interface

The front panel of the server VI remains hidden from the user during VI execution but can be displayed from the Servers in Use utility, accessible from the Engine Manager. The server can display general server status or other information on its front panel. 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.

Debugging and Testing Your Server

Testing Server Registration

You can use the Server Browser utility to view most of the information registered by your server. To launch the Server Browser utility, select **Tools»DSC Module»Advanced»Server Browser**. Select your server in the Servers list and click the **View Server Information** button to view the device and item information that has been registered by the server. If your server name does not appear in the list then the server is not registered in the currently selected CCDB.

Testing Server Operation

The DSC Module provides a utility, the Interactive Server Tester, that allows you to simulate the Tag Engine and run your server in LabVIEW where you can use LabVIEW debugging tools. You cannot properly test and debug your server without using this tool as the DSC Server Interface VIs do not execute correctly without either the Tag Engine or the Interactive Server tester running.

Note When you make a change to a server, you must force the Tag Engine to reload the server VI. You can do this by right-clicking the Tag Engine icon in the Windows task tray and selecting **Exit** or clicking **Stop** or **Exit** in the Engine Manager.

To launch the Interactive Server Tester, shown in Figure 2-6, choose **Tools»DSC Module»Advanced»Interactive Server Tester**.

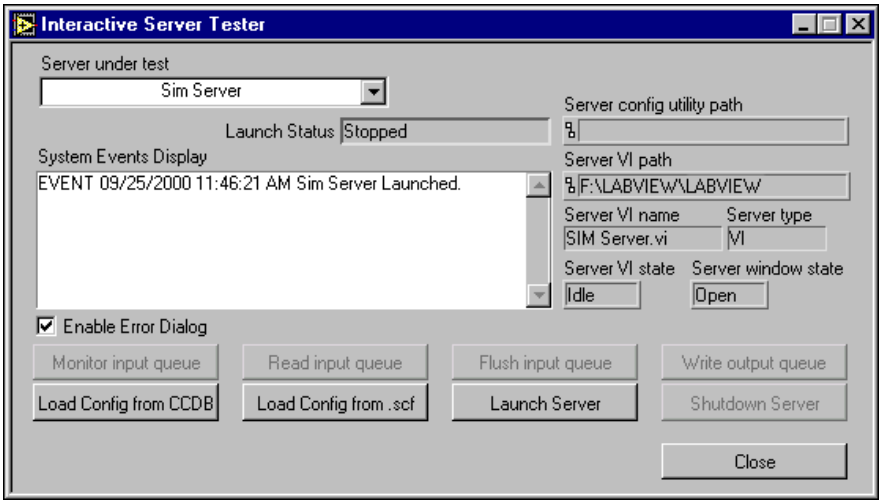

Figure 2-6. Interactive Server Tester Dialog Box

Choose the server you want to test in the **Server under test** listbox. The Interactive Server Tester shows the path to the server and other server information. You can then load the server item configuration for testing the server either from the CCDB or from a tag configuration (`.scf`) file. Click the **Load Config from CCDB** button to load all the server items that are currently registered in the CCDB. Click the **Load Config from .scf** button to select a `.scf` file that uses the server. When you select a `.scf` file, only the server items used by that file are loaded.

When the configuration is loaded, click the **Launch Server** button to launch and load the server. Click the **Monitor input queue** button to bring up a window that displays all items sent by the server to the DSC Module. Normally, you should click this button as soon as you launch the server if your server sends a lot of values to the DSC Module; otherwise, the input queue may overflow. Click the **Write output queue** button to bring up a window from which you can simulate writing item values to your server. The **Read input queue** button brings up a display that reads and displays all values currently in the input queue each time you click the **Read** button. If the Monitor Input Queue display is shown, it automatically reads all input values, so the Read Input Queue window may return little information. Click the **Flush input queue** button to clear out any values in the input queue.

When the server is open and running (the Interactive Server tester loads and shows the VI server top-level VI if it is not already open), you can use breakpoints and other debugging tools to monitor the server operation. When you are finished, click the **Shutdown Server** button to signal the server to stop execution. If the VI-based server operates correctly using the Interactive Server Tester, it should operate correctly when loaded and run by the Tag Engine.

Try using the SIM Server to familiarize yourself with the Interactive Server Utility.

Viewing the Server While Running in the Engine Process

When the server is running in the engine process, you cannot debug the server or view its diagram, but you can view the server front panel while it is running. Click the **View Servers in Use** button on the Engine Manager front panel to launch the Server Browser utility. Select your server in the server list—it appears with a diamond symbol next to the server name if the server is loaded, and the diamond is black if the server is running. Select your server and click the **Show Server User Interface** button to open the server and show the front panel while it is running. By adding indicators to the server front panel, you can monitor server operation while it is running in the engine. It is a good idea to minimize the amount of information displayed by the server while it is running so as not to use too much CPU bandwidth. You could use a user interface button on your server front panel that only displays server information when clicked.

Load and run a tag configuration file that uses the SIM server and then open the SIM server in the engine process to see how this works.

A

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at ni.com for technical support and professional services:

- **Support**—Online technical support resources include the following:
  - **Self-Help Resources**—For immediate answers and solutions, visit our extensive library of technical support resources available in English, Japanese, and Spanish at ni.com/support. These resources are available for most products at no cost to registered users and include software drivers and updates, a KnowledgeBase, product manuals, step-by-step troubleshooting wizards, conformity documentation, example code, tutorials and application notes, instrument drivers, discussion forums, a measurement glossary, and so on.
  - **Assisted Support Options**—Contact NI engineers and other measurement and automation professionals by visiting ni.com/support. Our online system helps you define your question and connects you to the experts by phone, discussion forum, or email.
- **Training**—Visit ni.com/custed for self-paced tutorials, videos, and interactive CDs. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, NI Alliance Program members can help. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.