LabVIEW™ Internet Toolkit User Guide

Version 6.0

The LabVIEW Internet Toolkit provides you with the ability to incorporate Internet capabilities into VIs. You can use LabVIEW to work with XML documents, build CGI programs and URLs, and send and receive data.

The Internet Toolkit includes the following components:

- **XML DOM Parser and VIs**—Create, access, and edit XML documents in LabVIEW.
- **CGI VIs**—Build CGI programs and manage client state information.
- **G Web Server**—Publish VIs to the Web and host CGI applications.
- **Online examples**—Learn how to build and use CGI VIs by reviewing online example VIs that you can run from the G Web Server.
- **URL Client VIs**—Parse and build URLs and download data from Internet servers.
- **FTP VIs**—Store and retrieve files from FTP (File Transfer Protocol) servers.
- **Telnet VIs**—Send and receive data using the Telnet protocol.

Refer to the *LabVIEW Help* for more information about the URL Client VIs, FTP VIs, and Telnet VIs.

Contents

January 2004
323747A-01

NATIONAL INSTRUMENTS™

Upgrading from Version 5.0

The Internet Toolkit version 6.0 contains the following changes from version 5.0:

- You can use the Internet Toolkit version 6.0 only with LabVIEW version 7.0 or later.
- You can select one of two different modes to run the G Web Server. In independent mode, the G Web Server runs on a port different from the one on which the LabVIEW Web Server runs. In shared-port mode, the G Web Server shares a port with the LabVIEW Web Server and handles only CGI requests, while the LabVIEW Web Server handles other requests. Refer to the *Configuring the G Web Server* section for more information about selecting modes for the G Web Server.
- You can use XML DOM Parser features to read, write, or process XML documents. Refer to the *Getting Started with the XML DOM Parser* section for more information about using XML features in LabVIEW.
- You can find the SMTP E-mail VIs on the **Functions** palette in the Full and Professional Development Systems of LabVIEW version 7.0. The filenames for these VI changed slightly. The Internet Toolkit includes compatibility libraries that automatically replace the version 5.0 VIs with the LabVIEW 7.0 VIs when you open a VI that contains the older version of the SMTP E-mail VIs.

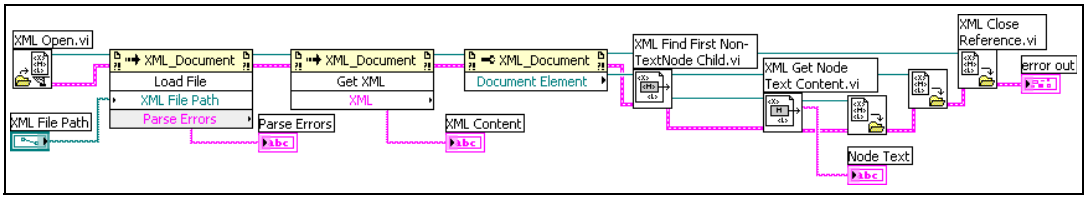
Getting Started with the XML DOM Parser

XML (Extensible Markup Language) is a platform-independent subset of SGML (Standard Generalized Markup Language) that you can use to store and exchange information. When you work with XML documents, you can use a parser to extract and manipulate data without having to translate the XML format directly. For example, the DOM (Document Object Model) Core specification defines a programming interface for creating, reading, and modifying XML documents. The DOM Core specification defines the properties and methods that an XML DOM Parser must support.

The Internet Toolkit includes an XML DOM Parser and a palette of XML DOM Parser VIs that you can use to read, write, or otherwise manipulate XML documents.

Using XML DOM Parser VIs

The Internet Toolkit XML DOM Parser VIs allow you to access a cross-platform XML DOM Parser in LabVIEW, as shown in the following example.



The XML Open VI opens an XML DOM Parser session and returns an XML Document reference. You can create an XML document, load an existing document into the XML DOM Parser, or configure document settings.

To manipulate an existing file, use the Load File method to load the XML document into memory. The XML document is accessible unless the XML DOM Parser encounters errors with the document.

Use the Get XML method to populate a string with the contents of the current XML object. In this example, the **XML Content** indicator displays the contents of the XML file.

The XML Close Reference VI closes the XML Document reference and returns any errors the VI encounters.

Note The XML DOM Parser ignores whitespaces only when it is performing validation. Otherwise, the parser includes whitespaces as child nodes.

Refer to the *LabVIEW Help* for more information about specific XML DOM Parser VIs, properties, and methods.

Validating XML Documents

You can configure the XML DOM Parser to determine if a specific XML document is valid, which means that the document complies with an external vocabulary. The external vocabulary can take the form of a DTD (Document Type Definition) or a schema.

A validating parser validates an XML document against a DTD or schema and reports invalid items that it finds. You must ensure that a specific document is the form and type that you expect. Using a validating parser eliminates the need to create custom validation code for each type of document.

The XML DOM Parser reports validation errors in the **Parse Errors** output of the Load File method.

Note You cannot validate a document that you currently have open and in memory. You must save and reload a document to validate it.

Refer to the `examples\internet\dom` directory for examples of VIs that use the XML DOM Parser to edit and validate XML documents.

Configuring the XML DOM Parser

You must configure all instances of the XML DOM Parser if you want to activate all possible features. You can configure the parser by setting properties on the XML_Document class.

Place a Property Node of class XML_Document on the block diagram and examine the available properties. The Property Node includes the following properties and methods:

- **Do Namespaces**—Allows you to enable or disable the XML DOM Parser namespace processing. The default is TRUE, which causes the XML DOM Parser to enforce its namespace specification constraints and rules.
- **Do Schema**—Allows you to enable or disable XML DOM Parser schema processing. The default is FALSE, which means the XML DOM Parser does not process any schema. If you set the property to TRUE, you also must enable namespace processing.
- **Load External DTD**—Allows you to enable or disable the loading of external DTDs. The default is TRUE, which means the XML DOM Parser allows you to load external DTDs. If you set the property to FALSE, the XML DOM Parser ignores external DTDs if you set the Validate on Load property to **Never**. The parser ignores this property if you set the Validate On Load property to **Always** or **Auto**.
- **Preserve Whitespace**—Allows you to specify if a validating parser includes ignorable whitespaces as text nodes. The default is TRUE, which adds ignorable whitespaces to the DOM tree as text nodes. If you set the method to FALSE, the XML DOM Parser discards all ignorable whitespace and does not add text nodes to the DOM tree.

Note The XML DOM Parser ignores whitespaces only when it is performing validation. Otherwise, the parser includes whitespaces as child nodes. For example, the first child of a tag is likely to be whitespace rather than the next element.

- **Schema Full Checking**—Allows you to set full schema constraint checking. The method takes effect only if you set the Validate on Load property to **Always** or **Auto**. The default is FALSE, which runs partial constraint checking. Full schema constraint checking can be time consuming or memory intensive.
- **Validate On Load**—Allows you to set the validation scheme that the XML DOM Parser uses. You can select one of the following enumerated values:
  - **Auto**—(Default) Turns on validation if the parser detects any internal or external DTD subset.
  - **Never**—Turns off validation.
  - **Always**—Turns on validation.

Using CGI in LabVIEW

CGI is a standard interface for external gateway programs to communicate with information servers, such as HTTP servers.

On the Web, when a client sends a request whose URL specifies a CGI application, the server decodes the request, loads the application, and executes the application. The application generates data and returns it to the server. The server then sends a reply that contains this data to the client, which displays the reply.

You can generate documents dynamically using CGI applications. This process can help you when the data in documents changes over time or when you generate the document according to user-supplied criteria.

You can supply parameters for CGI applications in the following ways, depending on the method that an HTML document uses to invoke a CGI application.

- `POST` method—Use only with HTML forms in which a user clicks a **Submit** button.
- `GET` method—Use with HTML links and some HTML forms.

When a user completes an HTML `POST` form and clicks the **Submit** button, the Web browser encodes the contents of the fields into an ampersand (`&`)-separated list of `name=value` parameter pairs and sends this string to the CGI application as the content of the `POST` request. For example, if

the form contains the fields **name** and **age** and the user enters `John Smith` and `27`, respectively, the Web browser sends the following string:

```
name=John%20Smith&age=27
```

The string `%20` in the name represents a space because `20` is the hexadecimal ASCII value of the space character.

When a user completes an HTML `GET` form and clicks a button or link, the Web browser encodes the contents into a question mark (`?`)-separated string and adds the string to the end of the CGI name. An HTML link with parameters explicitly specified usually invokes a CGI application through the `GET` method. For example, the following URL connects to the server `some.server.adr`, invokes the CGI application located in `/cgi-bin/example.vi`, and sends the **single parameter** parameter.

```
<http://some.server.adr/cgi-bin/example.vi?single%20parameter>
```

Similarly, the following URL invokes the CGI application with two parameters **name** and **age** with values `John Smith` and `27`, respectively.

```
<http://some.server.adr/cgi-bin/example.vi?name=John%20Smith&age=27>
```

Maintaining Client State Information with CGI VIs

HTTP is a stateless protocol. Each time a client wants a document from a server, the client must establish a new connection and send a request. The server receives the request, returns a reply, and closes the connection. The server does not maintain state information between individual connections.

You have several options for maintaining client state information across multiple connections. For example, you can insert information you collect into hidden fields of an HTML form, or you can use a cookie.

A cookie is a name-value pair that corresponds to information an HTTP server stores. Cookies can include information such as user preferences, login identification, and online purchasing information. Use cookies with HTTP servers to customize information that users receive and to keep track of files that users access on a Web site. You can use cookies to maintain information on the client side or on the server side.

CGI VIs store cookies as keyed arrays of name-value pairs: the name of the cookie, such as `userID`, and the cookie value, a number that the HTTP server uses to identify the information.

Using Client-Side Cookies

You can use client-side cookies to store state information on a client system. Client-side cookies are easy to use and persist even when the server is not running. However, not all browsers work with client-side cookies. Some users do not want servers to write information to their computers without their knowledge, and potential security risks can arise when hosts other than the one that stores the information can access client state information.

You can use the CGI Get Query Client Side Cookies VI, CGI Set Client Side Cookie VI, and CGI Set Multiple Client Side Cookies VI to read and specify client-side cookies. The G Web Server encodes in URL format the name and contents of each cookie that it specifies and decodes from URL format the name and contents of cookies it receives from a Web browser. Refer to the *LabVIEW Help* for more information about CGI VIs related to client-side cookies.

Using Server-Side Cookies

You can use server-side cookies to store state information about the server. Server-side cookies do not rely on browser software to work correctly, do not store data on client computers, do not transmit as much data as client-side cookies, and have fewer security risks because the server maintains security. However, servers maintain state information only for a certain time that the server specifies, after which the cookie automatically expires.

The G Web Server defines a server-side cookie as a cluster of two strings, cookie ID and address. The server uses the cookie ID string to identify a specific cookie. You can embed a cookie ID in the HTML document that a CGI application creates. The address string ensures that only clients with the same address can access a server-side cookie. If a specific client request creates a cookie, only that client can view and modify the cookie.

You can use the CGI Cookie VI, CGI Spool Cookie VI, CGI Add Params To Cookie VI, and CGI Build Cookie Document VI to work with server-side cookies in HTML documents. You can create, destroy, and modify server-side cookies and add and query information associated with cookies. You also can create HTTP connection-based cookies and documents that contain cookies. Refer to the *LabVIEW Help* for more information about CGI VIs related to server-side cookies.

Getting Started with the G Web Server

This section provides a step-by-step introduction to configuring and running the G Web Server. The G Web Server is an HTTP/1.0-compatible server that you can use to run applications on the Web. The G Web Server is a stand-alone VI that runs independently of other VIs that are running.

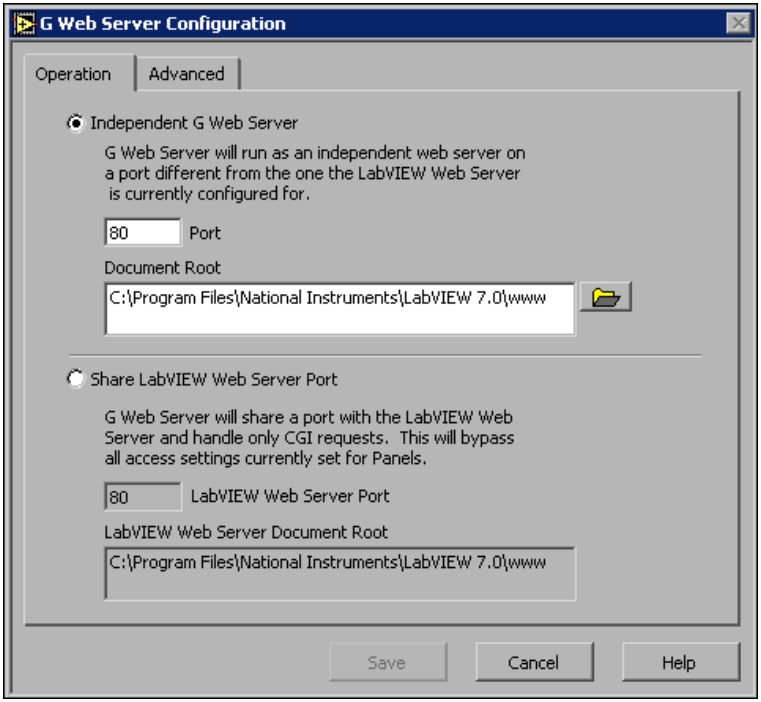
The G Web Server can perform the following tasks with VIs:

- **Run VIs**—You can use the G Web Server to load and run VIs that you create using the CGI VIs. You can run VIs on the Web without using a plug-in or run-time engine.
- **Work with CGI applications**—You can use CGI applications to create documents whose content frequently changes and to process queries and form requests. CGI applications run on HTTP servers and are different from applets you download and execute on client computers.
- **Integrate with other computers**—You can run the G Web Server on a development computer or integrate it into a stand-alone application.
- **Work on different platforms**—You can run the G Web Server on all platforms that work with LabVIEW version 7.0 or later. The CGI VIs that you use on any of these platforms run on any of the other platforms.
- **Implement security**—You can limit access by directory or by VI name. You can control access by the client address, the referring document, or through group and password files.
- **Publish front panel images**—You can use the G Web Server to publish the front panel of any VI in memory to the Web as a static or an animated image.

Configuring the G Web Server

To ensure the G Web Server runs correctly, you must set appropriate configuration options. Complete the following steps to configure the G Web Server.

1. Select **Tools»Internet Toolkit»G Web Server Configuration**. The **G Web Server Configuration** dialog box appears.
2. On the **Operation** tab, determine if you want to run the G Web Server in independent mode or in shared-port mode, as shown in the following dialog box.



In independent mode, the G Web Server runs on a port different from the one on which the LabVIEW Web Server runs. In shared-port mode, the G Web Server shares a port with the LabVIEW Web Server and handles only CGI requests, while the LabVIEW Web Server handles other requests.

3. Notice the **Port** text box, which specifies the TCP/IP port that the server is using. The default for HTTP is port 80. You might have to specify a different port if another HTTP server, including the LabVIEW Web Server, is already using port 80 on the computer or if you do not have permission to use reserved ports. If you use a non-default port, such as 8000, you must specify it on URLs that refer

to the server, such as `http://hostname:8000/index.htm`, where *hostname* is the computer name.

4. In the **Document Root** text box, specify the directory that contains the HTML document you want as the root or home page document. The default is the `www` directory, which contains the example home directory that the Internet Toolkit installs.
5. Click the **Advanced** tab. In the **Server Admin** text box, enter the email address you want to use. If the G Web Server encounters an error while retrieving a document, the server generates a document with the email address so viewers of the pages can alert you to the problem.
6. Remove the checkmark from the **Use DNS** checkbox if you do not have access to a DNS server. When you place a checkmark in the **Use DNS** checkbox, the server converts TCP/IP addresses, such as `130.164.140.14`, to their corresponding hostnames, such as `www.ni.com`. If you do not have access to a DNS server, looking up the name fails and significantly slows down the performance of the server.
7. Click the **Save** button to save the configuration settings.
8. Click the **Done** button to close the dialog box.

Note If you want to set configuration options other than those in the **G Web Server Configuration** dialog box, you must open the configuration file in a text editor to edit the options manually. The access configuration, server configuration, and server resource map configuration files are located in the `internet\http\conf` directory. Refer to the *LabVIEW Help* for more information about specific configuration files and directives.

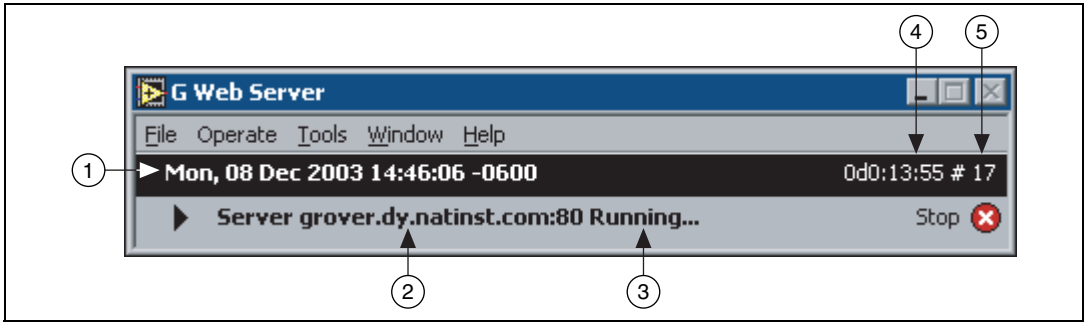
Running the G Web Server

Complete the following steps to run the G Web Server so you can examine its features.

1. Select **Tools»Internet Toolkit»Start G Web Server**. The **G Web Server** window opens.

Tip You also can run the G Web Server from the **Advanced** tab of the **G Web Server Configuration** dialog box. Click the **Start the G Web Server** button.

The TCP/IP hostname of the computer and the port you are using appear in the **G Web Server** window. You can view the **G Web Server** window in simple mode or detailed mode. The default is simple mode, shown in the following example, which uses less screen space and fewer system resources.



1	Current date, time, and timezone offset	4	Elapsed time the server has been running
2	Server name or IP address	5	Number of requests the server has handled
3	Status of server		

Refer to the *Displaying the G Web Server Window in Detailed Mode* section for information about viewing the **G Web Server** window in detailed mode.

2. Open a Web browser so you can experiment with the G Web Server.
3. Type the following string in the URL field of the Web browser:

   hostname`/.snap?HTTP+Server`

   Replace *hostname* with the name of the computer you are using, as displayed on the **G Web Server** window. If the hostname does not appear on the window, replace *hostname* with the word `localhost` or `127.0.0.1`, which is the IP address for the local host. A screenshot or animated image of the **G Web Server** window appears.

   In this example, the plus sign (`+`) replaces the whitespace in the name of the VI because a URL cannot contain spaces. You also can replace special characters with their hexadecimal value preceded by a percent (`%`) sign. For example, you would use `%20` for whitespace because `20` is the hexadecimal ASCII value of the space character. Otherwise, the name of the VI you observe should match the name in the VI menu bar. Each time you retype the URL or click the **Reload** or **Refresh** button on the browser, you can see a new image of the specified VI front panel.

Note URLs are not case sensitive. However, you must include filename extensions as part of the URL.

Viewing Online Examples

You can learn about the capabilities of the G Web Server by examining the online examples that the Internet Toolkit includes. Complete the following steps to access the default example Web page.

1. In the URL field of the Web browser, type the computer hostname, `localhost`, or `127.0.0.1`. The G Web Server Web page should appear. If you see something else, you must configure the document root directory properly. Refer to the *Configuring the G Web Server* section for more information about configuring the G Web Server.

Note To access the default example Web page when you are in shared-port mode, you must include `http://localhost/indexcgi.htm` in the URL field.

2. Click the **View CGI Examples** button to view a list of examples of G Web Server features. The examples demonstrate form processing through CGI VIs, static and animated panel images, password protection, and cookies.

To understand how the online examples work, you should be familiar with HTML code. To examine the HTML code for an example, select **View»Source** from the browser menu bar and search for the link, form element, or other feature that interests you. Many built-in examples invoke CGI VIs, for which the `HREF` for the link contains a VI name.

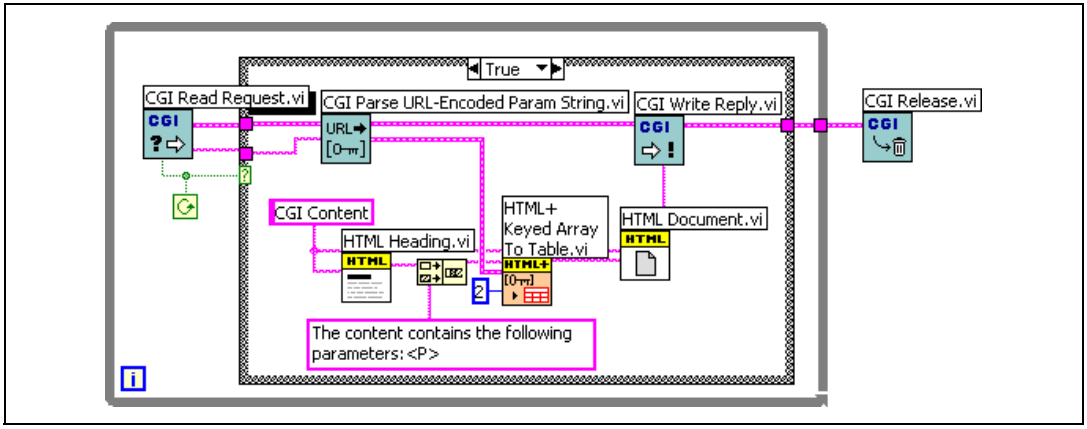
Examining Online CGI Examples

The Internet Toolkit includes online examples that you can access from the G Web Server home page. These examples demonstrate how you can use LabVIEW to create CGI applications that perform tasks over Web pages.

Complete the following steps to view an online example of a VI that uses CGI VIs.

1. From the G Web Server home page, click the **View CGI Examples** button.
2. Click the **CGI Basics** link.
3. Click the **CGI Call with Multiple Parameters (POST)** link to open the example.
4. Click the **Submit** button under the form to invoke the post_mlt VI, which uses the values of the form elements you selected to create an HTML document with a table that summarizes the values. The post_mlt VI is an example VI that contains CGI VIs as subVIs.

You can observe how an online example VI relates with a link or a **Submit** button by examining the HTML code for the example page. If you want to view the block diagrams of online example VIs, the VIs are located in the `www\cgi-bin\examples` directory. The following illustration displays the block diagram of the post_mlt VI.

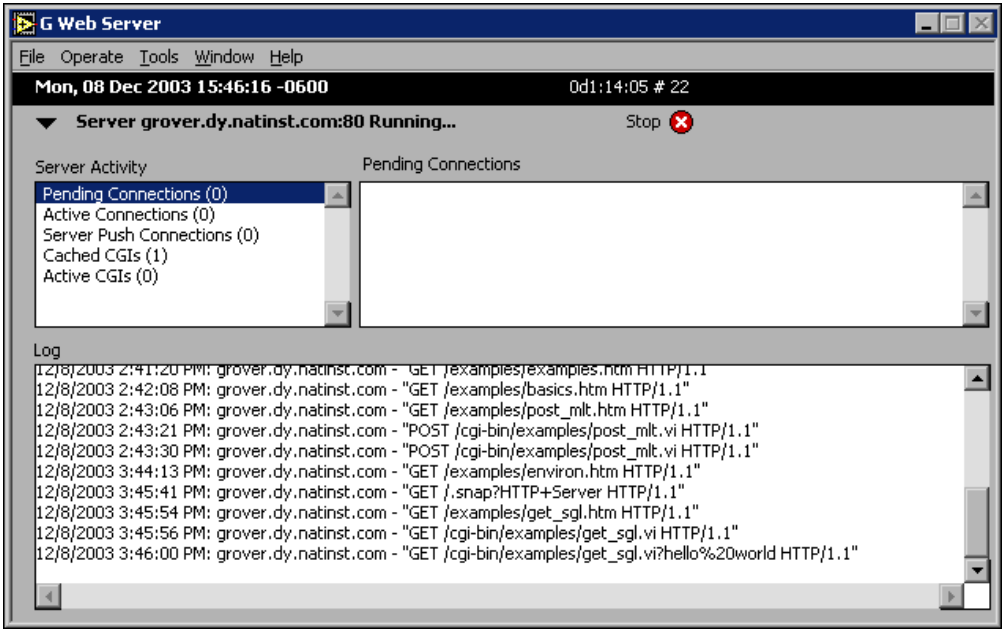


The CGI VIs that the post_mlt VI contains are subVIs that you must include in every CGI-related VI you create. The CGI Read Request VI obtains content strings that the client submits and provides that information to the rest of the block diagram in the form of a keyed array and a string. The CGI Write Reply VI sends a response to the client to display in the browser. Typically, this response comes in the form of HTML or a reference to an existing URL. The CGI Release VI frees the resources associated with this particular CGI call.

The CGI VIs in the post_mlt VI are specific to decoding the form elements that the CGI application received in this example using the `POST` method. These VIs are specific to building an HTML table to return to the client browser. Refer to the *LabVIEW Help* for more information about specific CGI VIs.

Displaying the G Web Server Window in Detailed Mode

The **G Web Server** window displays the G Web Server status information. You can view the window in simple mode or detailed mode. Click the black triangle next to the server name to switch the **G Web Server** window from simple mode to detailed mode, as shown in the following example.



In detailed mode, the **G Web Server** window displays the information from simple mode along with the following information:

- **Server Activity**—Displays information about different server aspects. When you select an item in the list, the text box on the right displays detailed information. You can select the following items:
  - **Pending Connections**—Indicates the number of connections that the server has accepted but not processed. Select this item to display addresses of the connections.
  - **Active Connections**—Indicates the number of open connections. Select this item to display addresses and requests of the connections.
  - **Server Push Connections**—Indicates the number of current server-push, or image animation, connections. Select this item to display addresses and requests of the server-push connections.
  - **Cached CGIs**—Indicates the number of CGI VIs currently in memory. Select this item to list CGI VI names, activity status, and the number of pending requests for each CGI VI.
  - **Active CGIs**—Indicates the number of CGI VIs currently processing requests.
- **Log**—Displays the most recent server requests and server error messages. Each request line consists of the date, time, remote system address, username or the symbol (-), and the request that the client sent. Error lines depend on the error condition, such as an invalid URL.

You can find more complete log information in the server log files, located in the `internet\http\logs` directory.

Using CGI VIs with the G Web Server

Servers and CGI applications communicate through environmental variables and through standard inputs and outputs. When an HTTP server executes a CGI application, it sends information using environmental variables. Refer to the *LabVIEW Help* for a list of environmental variables that LabVIEW supports.

Because LabVIEW does not work with standard HTTP input and output, a CGI VI receives the standard input data as a string when it receives a request and sends data that it generates as a string.

Publishing Front Panel Images with the G Web Server

You can use the G Web Server to publish images of front panels on the Web. You do not need to modify VIs to display images of their front panels. You can load static or animated front panel images. The G Web Server can generate images in JPEG or PNG image formats.

You can use the `.snap`, `.monitor`, and `.spool` URLs to publish images. If you use the G Web Server in shared-port mode, the LabVIEW Web Server handles `.snap` and `.monitor` requests and the G Web Server handles `.spool` requests. Refer to the *LabVIEW Help* for more information about publishing front panels to the Web.

Note Currently, only Netscape supports animated images of the front panel. Internet Explorer 5.0 or later does not support animated images, but it does periodically refresh the screen.

Where to Go from Here

The following documents and Web sites contain information you might find helpful.

- The *LabVIEW Help* includes information about VIs and functions and step-by-step instructions for using Internet Toolkit features and building stand-alone applications with Internet Toolkit components. Access the *LabVIEW Help* by selecting **Help»VI, Function, & How-To Help**. In LabVIEW 7.1 or later, navigate to the **Toolsets** book in the **Contents** tab for information specific to the Internet Toolkit. In LabVIEW 7.0, navigate to the **VI and Function Reference»Internet Toolkit VIs** book.

- *Using LabVIEW with TCP/IP and UDP* Application Note—Refer to the NI Developer Zone at `ni.com/zone` for new and updated Application Notes.
- `www.w3.org/Protocols`—Contains HTTP documentation and specifications.
- `www.w3.org/MarkUp`—Contains HTML documentation and specifications
- `hoohoo.ncsa.uiuc.edu/cgi`—Contains CGI documentation and specifications.
- `www.xml.org`—Contains information about XML standards and schemas.
- `www.w3.org/TR/REC-xml`—Contains information about XML 1.0 specifications.
- `www.w3.org/TR/DOM-Level-2-Core`—Contains information about DOM Level 2 Core specifications.
- `xml.apache.org`—Contains information about the Apache XML Project, which develops XML tools and standards.
- `xml.coverpages.org`—Provides resources about markup language standards.

Requests for Comments (RFC) documents form the basis for standard Internet protocols. Use a search engine to find the following RFC documents on the Web:

- File Transfer Protocol (FTP)—RFC 959
- HyperText Markup Language (HTML)—RFCs 1866 and 1942
- HyperText Transfer Protocol (HTTP)—RFC 1945
- Telnet Protocol—RFCs 854 and 855
- Multipurpose Internet Mail Extensions (MIME)—RFC 1521

323747A-01 Jan04