BridgeVIEW™ and LabVIEW™

Internet Developers Toolkit for G Reference Manual

June 1998 Edition
Part Number 321392B-01

Internet Support
E-mail: `support@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 (USA)
Tel: 512 795 8248
Fax: 512 794 5678

International Offices
Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 288 3336,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, Denmark 45 76 26 00, Finland 09 725 725 11,
France 01 48 14 24 24, Germany 089 741 31 30, Hong Kong 2645 3186, Israel 03 6120092, 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,
United Kingdom 01635 523545

National Instruments Corporate Headquarters
6504 Bridge Point Parkway Austin, Texas 78730-5039 USA Tel: 512 794 0100

© Copyright 1997, 1998 National Instruments Corporation. All rights reserved.

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

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

BridgeVIEW™, LabVIEW™, natinst.com™, National Instruments™, and The Software is the Instrument™ are trademarks of National Instruments Corporation.

Product and company names listed are trademarks or trade names of their respective companies.

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.

Contents

About This Manual

Organization of This Manual xi
Conventions Used in This Manual xii
Related Documentation xiii
Customer Communication xiii

Chapter 1
Introduction

What Is the Internet Developers Toolkit for G? 1-1
What Can I Do with the Toolkit? 1-1
Send E-mail and Data Files 1-2
Convert VIs to Internet Applications 1-2
Build CGI Programs in G 1-2

Chapter 2
Configuration

How Do I Configure My Toolkit? 2-1
How Do I Configure My Toolkit VIs? 2-2
[ftp] Configuration 2-2
[smtp] Configuration 2-2
[http] Configuration 2-3

Chapter 3
FTP VIs

What Is FTP? 3-1
What Are FTP VIs? 3-1
Low-Level VIs 3-2
Intermediate VIs 3-2
High-Level VIs 3-2
Which FTP VIs Can I Use? 3-3
Getting Started with FTP VIs 3-4

Chapter 4
E-Mail VIs

What Are E-Mail VIs? 4-1
Low-Level VIs 4-1
Intermediate VIs 4-2
High-Level VIs 4-2
What Is a Character Set? 4-2
US-ASCII Character Set 4-3
ISO Latin-1 Character Set 4-3
Macintosh Character Set 4-3
Transliteration 4-4
Which E-Mail VIs Can I Use? 4-5
Getting Started with SMTP VIs 4-5

Chapter 5
Telnet VIs

What Is a Telnet Connection? 5-1
Network Virtual Terminal 5-1
Negotiated Options 5-1
How Do I Use the Telnet VIs? 5-2
Which Telnet VIs Can I Use? 5-2
Getting Started with Telnet VIs 5-3

Chapter 6
URL VIs

What Is a URL? 6-1
Which URL VIs Can I Use? 6-2

Chapter 7
Getting Started with the G Web Server

What Is the G Web Server? 7-1
Why Use the G Web Server? 7-1
HTTP Server 7-2
How Do I Configure My G Web Server? 7-4
What Is HTML? 7-5
How Do I Get Started with the G Web Server? 7-6
Online Examples 7-8
CGI Basics 7-8

Chapter 8
Using the G Web Server

How Do My VIs Work with the G Web Server? 8-1
View Running VIs 8-1
Execute VIs 8-2
Embed Images 8-3
What URLs Can I Use with My Front Panel Images? 8-4
Front Panel Image Formats 8-4
Static Front Panel Image (.snap URL) 8-4
Syntax 8-5
Examples 8-6
Animated Front Panel Image (.monitor URL) 8-6
Syntax 8-6
Examples 8-8
Animated Front Panel Image (.spool URL) 8-8
Syntax 8-8
Examples 8-9
What Is the Common Gateway Interface? 8-10
CGI VIs with the HTTP Server 8-12
What Are CGI Utility VIs? 8-13
CGI Parameter VIs 8-14
File Path VIs 8-15
Network VIs 8-15
Panel Image VIs 8-16
Keyed Array VIs 8-16
HTML VIs 8-17
How Do I Maintain Client-State Information? 8-24
Client-Side Cookies 8-24
Using Client-Side Cookies 8-24
Client-Side Cookie Example 8-25
Client-Side Cookie VIs 8-25
Server-Side Cookies 8-25
Using Server-Side Cookies 8-26
Server-Side Cookie Example 8-26
Server-Side Cookie VIs 8-26
Advanced Configuration Options 8-28
Access Configuration 8-29
Configuration Directives 8-29
Server Configuration 8-30
Configuration Directives 8-30
Server Resource Map Configuration 8-31
Configuration Directives 8-31

Appendix A
Configuration Directives

Wildcard Expressions A-1
Access Configuration Directives A-2
Directory A-2
Panel A-3
AllowOverride A-4
AuthName A-5
AuthType A-5
AuthUserFile A-6
AuthGroupFile A-7
Limit A-8
Order A-9
Deny A-10
Allow A-11
Require A-12
Referer A-13
Satisfy A-14
OnDeny A-15
Server Configuration Directives A-16
Port A-16
UseDNS A-17
ServerAdmin A-17
ServerRoot A-18
ServerName A-19
StartServers A-19
TimeOut A-20
AccessConfig A-20
ResourceConfig A-21
TypesConfig A-22
CGICacheTime A-22
ErrorLog A-23
TransferLog A-24
AgentLog A-25
RefererLog A-26
LogOptions A-28
PanelImageType A-29
CheckPNGSupport A-30
PanelImageDepth A-30
PanelImageQuality A-31
PanelImageRefresh A-31
PanelImageCacheCompactTime A-32

ServerPushRefresh A-32
ServerPushLifespan A-33
ServerPushMaxConnections A-34
ServerPushMinTime A-34
Server Resource Map Configuration Directives A-36
DocumentRoot A-36
AccessFileName A-37
Redirect A-37
Alias A-38
ScriptAlias A-39
AddType A-39
AddEncoding A-40
DefaultType A-41
DirectoryIndex A-41
FancyIndexing A-42
DefaultIcon A-43
ReadmeName A-43
HeaderName A-44
AddIcon A-45
AddIconByType A-45
AddIconByEncoding A-46
IndexIgnore A-47
ErrorDocument A-48

Appendix B
Error Codes

Appendix C
Customer Communication

Glossary

Index

Figures

Figure 3-1. Read Contents of Remote File 3-4
Figure 3-2. Copy Contents of Remote File 3-5

Figure 4-1. How to Use the SMTP Send Message VI 4-6

Figure 5-1. Telnet VIs Contact Remote System and Log In 5-3

Figure 7-1. HTTP Server in Simple Mode 7-2
Figure 7-2. HTTP Server in Detailed Mode 7-3
Figure 7-3. Internet Toolkit Configuration Dialog Box 7-4
Figure 7-4. Example of TCP/IP Host Name 7-7
Figure 7-5. post_mlt VI Block Diagram 7-9

Figure 8-1. CGI Application Process 8-10
Figure 8-2. Structure of a CGI Application 8-13
Figure 8-3. HTML Block Diagram 8-18
Figure 8-4. HTML Table Block Diagram 8-18
Figure 8-5. HTML+ String Array to Table VI 8-19
Figure 8-6. HTML+ Monitor VI 8-19
Figure 8-7. HTML Generic Tag VI 8-20
Figure 8-8. HTML Document VI 8-20

Tables

Table 8-1. CGI Environment Variables 8-12

Table B-1. Error Codes B-1

About This Manual

The *Internet Developers Toolkit for G Reference Manual* describes the features, functions, and applications of the Internet Developers Toolkit for G. You can use this toolkit to send, receive, store, and publish information across the Internet using virtual instruments (VIs) you develop in the G programming environment.

Organization of This Manual

The *Internet Developers Toolkit for G Reference Manual* is organized as follows:

- Chapter 1, *Introduction*, introduces you to the Internet Developers Toolkit for G and describes its features and contents.
- Chapter 2, *Configuration*, describes how to configure the Internet Developers Toolkit for G.
- Chapter 3, *FTP VIs*, describes the *File Transfer Protocol (FTP)* VIs and their use in the Internet Developers Toolkit for G.
- Chapter 4, *E-Mail VIs*, describes the E-mail or *Simple Mail Transfer Protocol (SMTP)* VIs and their use in the Internet Developers Toolkit for G.
- Chapter 5, *Telnet VIs*, describes the *Telnet* VIs and their use in the Internet Developers Toolkit for G.
- Chapter 6, *URL VIs*, describes the *Universal Resource Locator (URL)* VIs and their use in the Internet Developers Toolkit for G.
- Chapter 7, *Getting Started with the G Web Server*, describes the G Web Server and how it works with the Internet Developers Toolkit for G and helps you get started using the G Web Server.
- Chapter 8, *Using the G Web Server*, explains how to work with the G Web Server.
- Appendix A, *Configuration Directives*, describes the configuration *directives* for the G Web Server and the use of wildcard expressions within the directives.
- Appendix B, *Error Codes*, lists the error codes the Internet Developers Toolkit for G VIs return, including the error number and a description.
- Appendix C, *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 manual, including abbreviations and acronyms.
- The *Index* contains an alphabetical list of key terms and topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

The following conventions are used in this manual:

<>

Angle brackets enclose the name of a key on the keyboard—for example, <shift>.

»

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.

☞

This icon to the left of bold italic text denotes a note, which alerts you to important information.

bold

Bold text denotes the names of menus, menu items, parameters, dialog boxes, dialog box buttons or options, icons, palettes, directives, or windows.

bold italic

Bold italic text denotes a note.

italic

Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text from which you supply the appropriate word or value, as in Windows 3.*x*.

`monospace`

Text in this font denotes text or characters you should literally 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 for statements and comments taken from programs.

`monospace italic`

Italic text in this font denotes that you must enter the appropriate words or values in the place of these items.

paths

Paths in this manual are denoted using backslashes (\) to separate drive names, directories, folders, and files.

Related Documentation

The following documents and World Wide Web sites contain information you might find helpful as you read this manual:

- Chapter 48, *TCP VIs*, in the *LabVIEW Function and VI Reference Manual*.
- Chapter 21, *TCP and UDP*, in the *LabVIEW User Manual*.
- *Network and Interapplication Communication* section of the LabVIEW *Online Reference*, available by selecting **Help»Online Reference**.
- `http://www.w3.org/Protocols/` for documentation and specifications of the *HyperText Transfer Protocol (HTTP)*
- `http://www.w3.org/MarkUp/` for documentation and specifications of the *HyperText Markup Language (HTML)*
- `http://hoohoo.ncsa.uiuc.edu/cgi/` for documentation and specifications of the *Common Gateway Interface (CGI)*
- `http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html` for a *Beginner's Guide to HTML*

Requests for Comments (RFC) documents form the basis for standard Internet *protocols*. You can use a search engine to find the following RFC documents at various Web and FTP sites on the Internet:

- File Transfer Protocol (FTP)—RFC 959
- HyperText Markup Language (HTML)—RFCs 1866 and 1942
- HyperText Transfer Protocol (HTTP)—RFC 1945
- Simple Mail Transfer Protocol (SMTP)—RFC 1869
- Telnet Protocol—RFCs 854 and 855
- Multipurpose Internet Mail Extensions (MIME)—RFC 1521

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 manual contains comment and configuration forms for you to complete. These forms are in Appendix C, *Customer Communication*, at the end of this manual.

1

Introduction

This chapter introduces you to the Internet Developers Toolkit for G and describes its features and contents.

What Is the Internet Developers Toolkit for G?

The Internet Developers Toolkit for G provides access to the Internet from within the G environment using virtual instruments (VIs) and servers. This toolkit consists of the following four main components:

- FTP VIs—Store and retrieve files from File Transfer Protocol (FTP) servers.
- E-mail VIs—Send messages with attached data to anyone on the Internet.
- Telnet VIs—Send and receive data using the Telnet protocol.
- URL VIs—Parse and build URLs and download data from Internet servers.
- G Web Server—Publishes your VIs on the World Wide Web and remotely controls your VIs from any Web *browser*.

What Can I Do with the Toolkit?

The Internet has created many new opportunities and uses for personal computers and workstations across every industry and application area. Currently, scientists and engineers perform research, publish conclusions, display data, and control source code versions across the Internet.

You can accomplish similar tasks by using your VIs in the G environment. Using the Internet Developers Toolkit for G, you can incorporate the following capabilities into your VI applications:

- Send e-mail or raw data files using your VIs.
- Convert your VIs into Internet-ready applications.
- Build *Common Gateway Interface (CGI)* programs in G.

Send E-mail and Data Files

With this toolkit, you can add e-mail and FTP capabilities to your VIs written in G. Using these *client-side* capabilities, you can send e-mail automatically when alarm conditions occur or send files or raw data to an FTP server from your application. You can use these capabilities for any remote field monitoring and control application that must pass data to a centralized reservoir or update operators on the status of the system.

Convert VIs to Internet Applications

You can convert your VIs into Internet-ready applications with this toolkit. You can use the G Web Server to view your VI *front panels* from any Web browser. With the built-in server, you can respond to multiple clients connecting to your program and continuously update the displays for each remote user. If you already are running a server, you can use the toolkit libraries programmatically to convert your VIs into image files for display within HyperText Markup Language (HTML) pages. You also can incorporate security levels into your server to limit access to your front panels and data by controlling access to your VIs based on a username and password or based on the user *Internet Protocol (IP)* address.

Build CGI Programs in G

You can build CGI VIs with this toolkit to use with your G Web Server. CGI VIs dynamically make decisions based on user input on a Web page. CGI VIs use a standard *Application Programming Interface* to receive a request from a remote user and take action, such as checking a username and password against a valid user list, returning a different HTML page to a user, making a computation and returning the results, or dynamically building an HTML page based on the data passed. This toolkit contains example CGI VIs and a template CGI VI to help you start receiving and sending requests and responses. You can use these examples to build your own CGIs using G.

2

Configuration

This chapter describes how to configure the Internet Developers Toolkit for G. Refer to the *Internet Developers Toolkit for G Version 5.0 Installation and Release Notes* for installation instructions.

How Do I Configure My Toolkit?

The Internet Developers Toolkit for G version 5.0 has been compiled for LabVIEW 5.0, but it also works with later versions of LabVIEW and BridgeVIEW 2.0. If you install this toolkit into a system other than LabVIEW 5.0, you must mass compile the `project\internet`, `vi.lib\addons\internet`, `examples\internet`, and `internet` directories before you can use the Internet Toolkit VIs.

To perform a mass compile on the contents of a given directory and its subdirectories, select **File»Mass Compile…**. Navigate to the directory you want and click the **Select Cur Dir** button.

The G Web Server includes an excellent example homepage that demonstrates most of the features of the server. To access this page, confirm that it is set as your G Web Server default homepage.

To verify this setting, select **Project»Internet Toolkit»Internet Toolkit Configuration…** and choose **HTTP (Web Server)** from the drop-down menu in the **Internet Toolkit Configuration** dialog box. The document root directory should be set to *`LabVIEW directory`*`\internet\home`.

Using the **Internet Toolkit Configuration** dialog box, you also can control things such as the default mail server for the SMTP VIs, the default password to use when opening an anonymous FTP session, the password lists and access control for the pages to which your server provides access, and more.

You can review and edit all these configuration settings by using a simple text editor. Refer to the following *How Do I Configure My Toolkit VIs?* section for more information.

How Do I Configure My Toolkit VIs?

You can leave some parameter settings in the Internet Toolkit VIs unwired. If left unwired, they use the default values configured in the `internet.ini` file, located in the `internet` directory.

The `internet.ini` file contains the following three configuration sections:

- `[ftp]`
- `[smtp]`
- `[http]`

[ftp] Configuration

Note ***You also can perform FTP configuration by selecting* Project»Internet Toolkit»Internet Toolkit Configuration… *and choosing* FTP (File Transfer) *from the drop-down menu in the* Internet Toolkit Configuration *dialog box.***

In the FTP configuration section, specify the following setting for your FTP VIs:

Set the password used to connect to an FTP site.
`AnonymousPassword=email_address`

For example
`AnonymousPassword=joe@some.address.com`

[smtp] Configuration

Note ***You also can perform SMTP configuration by selecting* Project»Internet Toolkit»Internet Toolkit Configuration… *and choosing* SMTP (Email) *from the drop-down menu in the* Internet Toolkit Configuration *dialog box.***

In the Simple Mail Transfer Protocol (SMTP) configuration section, specify the following settings for your SMTP (E-mail) VIs:

- Set the return address used by outgoing mail.
  `ReturnAddress=email_address`

  For example
  `ReturnAddress=joe@some.address.com <Joe Smith>`
- Set the return subject used by outgoing mail.
  `DefaultSubject=subject`

For example
`DefaultSubject=Control System Message`

- Set the server used by the SMTP VIs.
  `Server=internet_smtp_server`

  For example
  `Server=smtp.foo.com`

- Set the character set used by the SMTP VIs.
  `CharSet=character_set`

  For example
  `CharSet=iso-8859-1`

- Set the character mapping settings for converting between virtual and real character sets. The character map settings are a comma-separated list of triplets, three terms listed together in the following form: `<virtual charset> <charset> <mapping file>`.
  `Transliterate=translit_specs`

  For example
  `Transliterate=MacRoman iso-8859-1 macroman.trl`

  This example changes the contents of a text message whose character set is `MacRoman` to a message whose character set is `iso-8859-1` by mapping it with the mapping table stored in the file `macroman.trl`. Refer to the *Transliteration* section in Chapter 4, *E-Mail VIs*, for more information.

[http] Configuration

Note ***You complete most of the configuration for the G Web Server through separate server configuration files. Refer to the** How Do I Configure My G Web Server? **section in Chapter 7**, Getting Started with the G Web Server**, for more information on these specific files.***

In the HyperText Transfer Protocol (HTTP) configuration section, specify the following setting for your HTTP (G Web) Server:

Set the location of the HTTP Server configuration file.
`lvhttp.cfg=file_path`

For example
`lvhttp.cfg=c:\labview\internet\http\conf\lvhttp.cfg`

3

FTP VIs

This chapter describes the File Transfer Protocol (FTP) VIs and their use in the Internet Developers Toolkit for G.

What Is FTP?

You can use the FTP to transfer files between remote computers on the Internet. With FTP, you can download files and directory listings from the server, upload files, or create and delete directories on the server. To transfer files to or from a remote host, the host must use FTP server software and the local computer must use FTP client software to connect to the remote server. Most FTP servers require user authentication to give remote users access to files stored on the server.

When you transfer data between an FTP client and an FTP server, you must establish a second connection. You can make this connection an *active FTP mode* or a *passive FTP mode* connection. In the default active mode, the server connects to a data port the client specifies. In the passive mode, the client connects to the server data port. All FTP servers implement the active mode, which might not work for clients behind firewalls that block all external connection requests. Some older FTP servers might not implement the passive mode, which can work for clients behind firewalls.

What Are FTP VIs?

The FTP VIs implement an FTP client in G. With these VIs, you can transfer files programmatically to and from FTP servers on the Internet in *Image mode* or American Standard Code for Information Interchange (ASCII) mode. If you pass an empty string to the **username** parameter of an FTP VI, you connect as user `anonymous`. If you pass an empty string for the **password**, the VI uses the `AnonymousPassword`, which is your e-mail address, specified in the `[ftp]` section of the `internet.ini` configuration file.

You can specify whether data should be transferred in Image or ASCII mode with the VIs that transfer data, such as all the high-level VIs and some of the intermediate and low-level VIs. You also can determine the way to establish the data connection used for transferring data.

When you transfer text files, use ASCII mode. In this mode, the FTP VIs convert line-terminating characters to and from the Internet standard `CRLF`. Use Image mode for binary files, such as when transferring data log files or VIs. In Image mode, the transferred data is not modified. The FTP Browser example illustrates how to transfer both types of files. You can access this and other examples by selecting **Project»Internet Toolkit»Internet Toolkit Examples…**.

The FTP VIs are divided into three categories:

- Low-level VIs—Implement the commands in the FTP specification.
- Intermediate VIs—Perform FTP tasks consisting of low-level FTP commands.
- High-level VIs—Perform file transfer tasks when the filenames are specified.

Low-Level VIs

You can use the low-level VIs if you want to customize FTP for a specific task or if you want to use some of the other FTP commands. The low-level VIs directly implement the commands in the FTP specification, such as creating, listing, or deleting directories.

Intermediate VIs

You also can use the intermediate VIs if you want to customize FTP for a specific task or if you want to use some of the other FTP commands. The intermediate-level VIs are built on top of the low-level VIs and implement common tasks consisting of a series of low-level commands. The intermediate-level VIs also include VIs that open and close connections to FTP servers.

High-Level VIs

You can use the high-level VIs to complete entire tasks for you. First, you specify the path on the remote FTP server and the user authentication and local file information. Then, the VIs connect to the remote host, transfer data between your computer and the FTP server, and close the connection. With the high-level VIs, you can transfer one or more files directly to and from memory or files on disk.

For the input, each high-level VI uses the address of the FTP server, a username and password, remote paths to the location on the server, and local paths or data that include information about where the local data should be.

Which FTP VIs Can I Use?

You can use the following VIs to transfer files between a remote FTP server and a local system. With these VIs, you can read data from a file, store data to a file, or pass the data as a string parameter, or buffer. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»FTP VIs**. For more information about these VIs, right-click on the VI in the *block diagram* and select **Online Help** from the VI pop-up menu.

The following VIs are high-level VIs. Refer to the *Internet Toolkit VIs* online help for more information on the low-level and intermediate VIs.

- FTP Get Buffer VI—Connects to an FTP server and returns the contents of a single file.
- FTP Get File VI—Connects to an FTP server and copies a single file to the local system.
- FTP Get Multiple Buffers VI—Connects to an FTP server and returns the contents of a list of files.
- FTP Get Multiple Files VI—Connects to an FTP server and copies a list of files to the local system.
- FTP Get Multiple Files and Buffers VI—Connects to an FTP server and copies a list of files, saving them on the local system or returning the contents.
- FTP Put Buffer VI—Connects to an FTP server and stores the contents of a single buffer.
- FTP Put File VI—Connects to an FTP server and copies the contents of a single file from the local system.
- FTP Put Multiple Buffers VI—Connects to an FTP server and stores the contents of multiple buffers.

- FTP Put Multiple Files VI—Connects to an FTP server and copies the contents of multiple files from the local system.
- FTP Put Multiple Files and Buffers VI—Connects to an FTP server and copies the contents of multiple files or buffers from the local system.

The default settings for the FTP VIs are configured in the `internet/internet.ini` file.

Getting Started with FTP VIs

You can access the FTP VIs on the **Functions** palette by selecting **Internet Toolkit»FTP VIs**.

Using the FTP VIs is fairly straightforward. The high-level VIs are arranged in two rows: one Get row to bring remote files to your computer and one Put row to send local files to a remote computer. The local source or destination for data can be a file or a buffer.

When you choose a file, you must specify a local path to control where the remote data is stored (Get) or from which file local data is sent (Put). When you choose buffer, you store the remote data in a LabVIEW string indicator (Get) or send the contents of a string control to the remote system (Put).

Figure 3-1 illustrates the use of the high-level FTP Get Buffer VI to read the contents of a remote file into a LabVIEW string indicator.

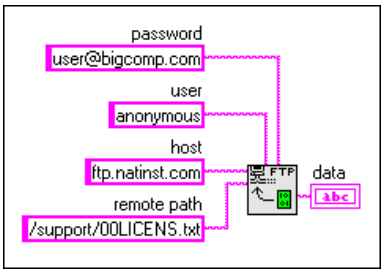

Figure 3-1. Read Contents of Remote File

Figure 3-2 shows an example of how to obtain the same information from the remote file but instead copy that information into a local file.

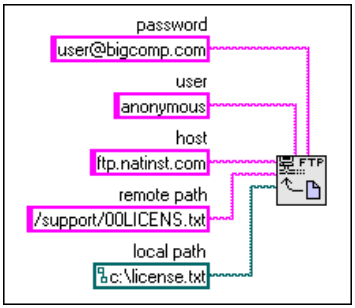

Figure 3-2. Copy Contents of Remote File

You must use UNIX-style paths, or forward slashes, when you specify the location of the remote file. The path style for the local destination depends on your operating system. Figure 3-2 uses a DOS-style path, or backslashes, because the local computer was running Windows.

4

E-Mail VIs

This chapter describes the E-mail or Simple Mail Transfer Protocol (SMTP) VIs and their use in the Internet Developers Toolkit for G.

What Are E-Mail VIs?

You can use the E-mail VIs to send electronic mail, including attached data and files, using SMTP. When you use the E-mail VIs, your messages are encoded using the *Multipurpose Internet Mail Extensions (MIME)* format. In this format, you can send multiple documents, including binary data files, within an e-mail message. You also can describe properties of the individual attachments, for example, what character set a text document uses.

To send e-mail on the Internet, you must have the e-mail address of one or more recipients and the Internet address of an SMTP server. The E-mail VIs open a connection to the server and send the server commands that describe the recipients and the contents of the e-mail message. The server sends the message to the individual recipients or forwards it to other SMTP servers. You can access examples that illustrate how to use the E-mail VIs by selecting **Project»Internet Toolkit»Internet Toolkit Examples…**.

The E-mail VIs are divided into three categories:

- Low-level VIs—Implement the individual SMTP commands.
- Intermediate VIs—Perform tailored e-mail commands.
- High-level VIs—Send your completed messages for you.

Low-Level VIs

You can use the low-level VIs to access the actual commands SMTP defines. Using these VIs, you can circumvent the formatting the intermediate VIs perform, or you can use commands that are not part of SMTP but that work with your server.

Intermediate VIs

You can use the intermediate VIs to tailor the contents of an e-mail message and to send multiple e-mail messages over one connection. These VIs can open and close connections to an SMTP server and encode message parts to MIME format. The intermediate VIs are built on top of the low-level VIs.

You can leave several input parameters in the intermediate VIs unwired. If left unwired, the parameters use the default values specified in the `internet.ini` configuration file. Refer to the *[smtp] Configuration* section in Chapter 2, *Configuration*, for more information on these parameters.

High-Level VIs

You can use the high-level VIs to specify the sender, recipients, message, and attachments to your e-mail. These VIs open a connection to an SMTP server, send the message, encode it in MIME format, and close the connection. The high-level VIs are easy to use and cover many e-mail tasks.

Each high-level VI uses the addresses of the recipients, a subject, a message, and attachments as the input. You also can specify the sender, the SMTP server to use, and the character set the message and text attachments use. The high-level VIs are built on top of the intermediate VIs.

You can leave several input parameters in the high-level VIs unwired. If left unwired, the parameters use the default values specified in the `internet.ini` configuration file. Refer to the *[smtp] Configuration* section in Chapter 2, *Configuration*, for more information on these parameters.

What Is a Character Set?

The high-level VIs and some of the intermediate VIs contain a **character set** input parameter. This parameter specifies the character set to use in the text of the e-mail message or attachment. A character set describes the mapping between *characters* and their *character codes*.

A character is a basic unit of written languages: a letter, a number, a punctuation mark, or, in some languages, an entire word. Modifications of letters, such as capitalization or accent marks, make that letter a separate character. For example, the characters `O`, `o`, ö, and ô are all different characters.

A character code is a number used to represent a given character. Because computers deal only with numbers, they must associate a character with a number to operate on the character.

A character set is the relationship between characters and the numbers that represent them in the computer. For example, in the American Standard Code for Information Interchange (ASCII) mode, the character codes for `A`, `B`, and `C`, are `65`, `66`, and `67`, respectively.

US-ASCII Character Set

The most widely used character set on the Internet is the US-ASCII or ASCII character set. Most Internet programs, including e-mail applications, use this character set as the default and do not work with any other sets. The ASCII character set describes all the letters and most punctuation marks used in the English language, for a total of 128 characters. Most other character sets are extensions of ASCII.

Using the ASCII character set might not be sufficient because many languages require characters not found in ASCII. For example, you cannot write the German word `Müller` using ASCII because the character ü is not defined in the ASCII character set.

ISO Latin-1 Character Set

Because many languages require characters not found in ASCII, countries that use these languages created new character sets. Most of these character sets contain the first 128 character codes as ASCII and define the next 128 character codes to describe characters needed in that language. Some of these character sets use different character codes to describe the same characters. This can cause problems when one character set displays text written in another character set. To solve this problem, some standard character sets are used. One widely used standard character set is ISO Latin-1, also known as ISO-8859-1. This character set describes characters used in most western European languages and is understood by most e-mail applications that deal with those languages.

Macintosh Character Set

Apple Computer, Inc. developed its own extended character set before ISO Latin-1 was defined. The Macintosh character set is based on ASCII but uses a different set of upper 128 character codes than ISO Latin-1. Because of this, e-mail that contains accented characters written in the Macintosh character set appears incorrectly in e-mail applications that expect ISO Latin-1 text. To solve this problem, Macintosh e-mail

applications convert text to ISO Latin-1 before mailing it. When another Macintosh e-mail application receives text designated as using the ISO Latin-1 character set, it converts the message to the Macintosh character set.

Transliteration

Using the E-mail VIs, you can specify character sets that map text to another character set before sending the text. The mapping of characters is called *transliteration*. In the `[smtp]` section of the `internet.ini` configuration file, you can specify which transliterations the E-mail VIs can work with. A transliteration is defined by a *virtual character set*, a *target character set*, and a transliteration file. The default configuration file contains one defined transliteration, `[MacRoman iso-8859-1 macroman.trl]`. When you send a message and specify that the character set is `MacRoman`, the E-mail VIs transliterate the text and send it out as character set `iso-8859-1` (ISO Latin-1).

The transliteration file is a binary file of 256 bytes. The value of each entry contains the new character code of the mapped character. Entries with a value the same as their index do not modify that particular character. In other words, the *identity transliteration file* is a file that consists of values 0, 1, 2, 3, …, 255. It does not change the transliterated text. For example, a transliteration file that changes ASCII letters to uppercase is the same as the identity transliteration file except that entries representing lowercase characters (97, 98, …, 122) contain the character codes of the equivalent uppercase characters (65, 66, …, 90). If the name of this file is `asciiup.trl`, the transliteration entry in the configuration file might be `ASCIIUpper us-ascii asciiup.trl`.

When a transliteration specifies another transliteration as the target character set, the mappings are applied in the expected order. For example, if the transliteration entry is [`MacRoman iso-8859-1 macroman.trl, MacRomanUp MacRoman asciiup.trl`], the character set `MacRomanUp` first changes all ASCII characters in the text to uppercase using `asciiup.trl` and then changes the text to ISO Latin-1 using `macroman.trl`.

Which E-Mail VIs Can I Use?

You can use the following VIs to send e-mail with or without attachments to one or more users. The attachments can come directly from a string or can be a file on a local disk. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»Email VIs**. For more information about these VIs, right-click on the VI in the block diagram and select **Online Help** from the VI pop-up menu.

The following VIs are high-level VIs. Refer to the *Internet Toolkit VIs* online help for more information on the low-level and intermediate VIs.

- SMTP Send Message VI—Sends text e-mail message to a list of recipients.
- SMTP Send Data VI—Sends an e-mail message with attached data to a list of recipients.
- SMTP Send File VI—Sends an e-mail message with an attached file to a list of recipients.
- SMTP Send Multiple Attachments VI—Sends an e-mail message with multiple data and file attachments to a list of recipients.
- SMTP Send Message (Small) VI—Sends a text e-mail message to a single recipient.

The default settings for the E-mail (SMTP) VIs are configured in the `internet/internet.ini` file.

Getting Started with SMTP VIs

You can access the SMTP VIs on the **Functions** palette by selecting **Internet Toolkit»Email VIs**.

One important thing to remember regarding the SMTP VIs is that you must specify a valid mail server either in the `internet\internet.ini` file or by wiring a valid input into the mail server terminal of a particular mail VI every time you use it. The mail server must be the host name or IP address of an external server computer that is willing to service requests from the computer running the Internet Toolkit. If you are uncertain about the appropriate mail server to use, contact your network administrator to get the name of a valid server.

Figure 4-1 shows an example of how to use the SMTP Send Message VI on a LabVIEW block diagram to send a message programmatically. In this example, the inputs specified are the mail server, the intended recipient of the message, the subject, and the content of the message.

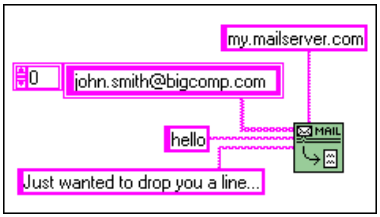

Figure 4-1. How to Use the SMTP Send Message VI

You can use other SMTP VIs to send messages with data attachments and/or file attachments by specifying the data buffer (string control) or filename to attach.

5

Telnet VIs

This chapter describes the Telnet VIs and their use in the Internet Developers Toolkit for G.

What Is a Telnet Connection?

A Telnet connection is a *Transmission Control Protocol (TCP)* connection used to transmit data that contains Telnet control information. The Telnet protocol provides a general, bi-directional, 8-bit byte-oriented communications facility. The Telnet protocol also provides a standard method of connecting terminal devices and terminal-oriented processes to each other.

The Telnet protocol is built on the following two ideas:

- The concept of a *Network Virtual Terminal (NVT)*
- The principle of negotiated options

Network Virtual Terminal

When a Telnet connection is first established, each end is assumed to originate and terminate at an NVT. An NVT is an imaginary device that provides a standard, networkwide, intermediate representation of a canonical terminal. An NVT eliminates the need for server and user hosts to keep information about the characteristics of each terminal and terminal handling conventions. All hosts, the server and the user, map their local device characteristics and conventions so they appear to be dealing with an NVT over the network. Each host assumes a similar mapping by the other party.

Negotiated Options

The principle of negotiated options recognizes the fact that many hosts want to provide services in addition to those available within an NVT and that many users have sophisticated terminals and want elaborate, rather than minimal, services. To meet this need, the Telnet protocol provides various options that allow a server and a user to agree on a set of

conventions for their Telnet connection. These options are independent of but structured within the Telnet protocol.

How Do I Use the Telnet VIs?

The Telnet VIs form a transparent layer between your application and the Telnet protocol. These VIs implement the most basic NVT by interpreting and rejecting all option negotiation requests. With the Telnet VIs, your applications can communicate with remote applications using the Telnet protocol as if it were a regular TCP connection. You can use the Telnet VIs to log on programmatically to a remote shell and execute commands or to control instruments that use the Telnet protocol.

Telnet is a protocol used to connect to and log onto a remote computer. The remote computer must be running a Telnet server for these VIs to work. When you connect to a remote computer via Telnet, you must always supply a username and a password to gain access. In most cases, anonymous users do not have access. You must have a valid account on the remote computer to connect.

The Telnet Line Client example illustrates how to use the Telnet VIs. You can access this and other examples by selecting **Project»Internet Toolkit»Internet Toolkit Examples…**.

Which Telnet VIs Can I Use?

You can use the following Telnet VIs to connect your application with the Telnet protocol using the basic NVTs. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»Telnet VIs**. For more information about these VIs, right-click on the VI in the block diagram and select **Online Help** from the VI pop-up menu.

- Telnet Open Connection VI—Attempts to open a Telnet connection with the specified address and port.
- Telnet Read VI—Reads data from the Telnet connection, filtering out Telnet control characters.
- Telnet Write VI—Writes the string data to the specified Telnet connection.
- Telnet Play Script VI—Executes script on the specified Telnet connection.
- Telnet Close Connection VI—Closes the Telnet connection.

- Telnet Listen VI—Creates a listener and waits for an accepted Telnet connection at the specified port.
- Telnet Wait On Listener VI—Waits for an accepted Telnet connection using the specified listener.

Getting Started with Telnet VIs

You can access the Telnet VIs on the **Functions** palette by selecting **Internet Toolkit»Telnet VIs**.

The Internet Developers Toolkit for G Telnet VIs enable programmatic Telnet sessions, in which commands in the form of strings are sent to the remote computer after a connection has been established. Figure 5-1 gives an example of how to use the Telnet Open Connection VI and the Telnet Write VI to contact a remote system and begin the login process by sending a valid username. The Telnet Close Connection VI ultimately closes the session.

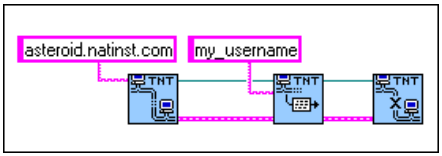

Figure 5-1. Telnet VIs Contact Remote System and Log In

The Internet Toolkit includes a Telnet Client VI that enables a user to establish a remote connection and enter commands interactively to the remote host. To start the Telnet Client VI, select **Project»Internet Toolkit»Internet Toolkit Examples** and open the Telnet Line Client example.

6

URL VIs

This chapter describes the Universal Resource Locators (URL) VIs and their use in the Internet Developers Toolkit for G.

What Is a URL?

The URLs are the standard method for describing resources on the Internet. A fully formed Internet URL consists of the following format:

`protocol://user:password@host:port/virtual-path`

You can omit the authentication (`user:password@`) and the Transmission Control Protocol/Internet Protocol (TCP/IP) port specification (`:port`), which instruct the browser to supply the username and password and the default port for the specified protocol. Some protocols, such as `mailto`, do not use a virtual path but instead require a username.

The following list provides some example URLs and their functions:

- `http://www.natinst.com`—Uses the `http` protocol to access the default document from `www.natinst.com`.
- `http://www.natinst.com/labview`—Uses the `http` protocol to access the `/labview` document from `www.natinst.com`.
- `http://some.comp.adr:8080/info.htm`—Uses the `http` protocol to connect to port `8080` at `some.comp.adr` and retrieve the document `/info.htm`.
- `ftp://john:smith@ftp.some.addr/pub/readme.txt`—Uses the `ftp` protocol to log on to `ftp.some.addr` as user `john` with password `smith` and retrieve the file `pub/readme.txt`.
- `mailto:g.internet@natinst.com`—Specifies the e-mail address `g.internet@natinst.com`.

Which URL VIs Can I Use?

You can use the following VIs to build and *parse* URLs and to retrieve documents by specifying their URL. Some of the URL VIs also work with URLs that are not fully specified. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»URL VIs**. For more information about these VIs, right-click on the VI in the block diagram and select **Online Help** from the VI pop-up menu.

- Build URL VI—Constructs a URL string from a protocol, host, user password, port, and virtual path.
- Parse URL VI—Parses a fully or partially formed URL and returns its components.
- URL Get HTTP Document VI—Retrieves a document specified by a URL using the HTTP protocol and following redirections to other HTTP URLs.
- URL Get Gopher Document VI—Retrieves a document specified by a URL using the Gopher protocol.
- URL Get FTP Document VI—Retrieves a document specified by a URL using the FTP protocol.
- URL Get Document VI—Retrieves a document specified by a URL using the HTTP, Gopher, or FTP protocol as specified in the URL and following redirections to other URLs.

7

Getting Started with the G Web Server

This chapter describes the G Web Server and how it works with the Internet Developers Toolkit for G and helps you get started using the G Web Server.

What Is the G Web Server?

The G Web Server is an HTTP/1.0-compatible server for making *HyperText Markup Language (HTML)* and other documents available on the Internet and for connecting VIs to the World Wide Web. With the G Web Server, you can publish your VIs on the World Wide Web and control your VIs from any Web browser.

The G Web Server, a type of HyperText Transfer Protocol (HTTP) server, is a standalone VI that executes independently of other VIs that are running. You can start the server by selecting **Project»Internet Toolkit»Start HTTP Server…** or by using the HTTP Server Control VI. The HTTP Server Control VI programmatically tests the state of the G Web Server and loads, starts or stops, and unloads the server VIs. For more information about this VI, right-click on the VI in the block diagram and select **Online Help** from the VI pop-up menu.

Note ***To start working with the G Web Server immediately, refer to the*** *How Do I Get Started with the G Web Server?* ***section later in this chapter.***

Why Use the G Web Server?

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

- Publishes virtual instruments on the Web—You can publish static or animated images of your VIs on the Word Wide Web without modifying your VIs.
- Works with CGI virtual instruments—You can develop CGI VIs that execute dynamically when a browser requests them. You can use your CGI VIs with all platforms that work with G.

- Works with CGI animations—You can work with *server-push animations* CGI VIs generate.
- Works with imagemap files—You can work with imagemap files without an external CGI application.
- Integrates with other systems—You can run the G Web Server on a development system or integrate it into your executable.
- Works on different platforms—You can run the G Web Server on all platforms that work with BridgeVIEW 2.0 or LabVIEW 5.0 or later. CGI VIs written on any of these platforms run on any of the other platforms.
- Implements security—You can limit access by directory or VI name. You can control access by the client address, the referring document, or through group and password files.

HTTP Server

The front panel of the HTTP Server displays the G Web Server status information. You can view the front panel in simple mode or detailed mode. Simple mode uses less screen space and is more efficient using system resources. Figure 7-1 shows the HTTP Server in simple mode.

Figure 7-1. HTTP Server in Simple Mode

The text on the left side of the black bar displays the current date, time, and time zone offset.

The text on the right side of the black bar displays the elapsed time since you started the server. In Figure 7-1, the server started 5 days, 17 hours, 27 minutes, and 50 seconds ago. The black bar also displays the number of requests the server handled during this time: 1,544 requests.

The **Stop** button below the black bar stops the HTTP Server, closes all open connections, and unloads cached Common Gateway Interface (CGI) VIs from memory.

The bold text below the black bar displays the current status of the server. Click the black triangle to change the front panel to detailed mode.

The detailed mode, shown in Figure 7-2, displays information about the current state of connections, current front panel animations, active or cached CGI VIs, and a current log of client requests.

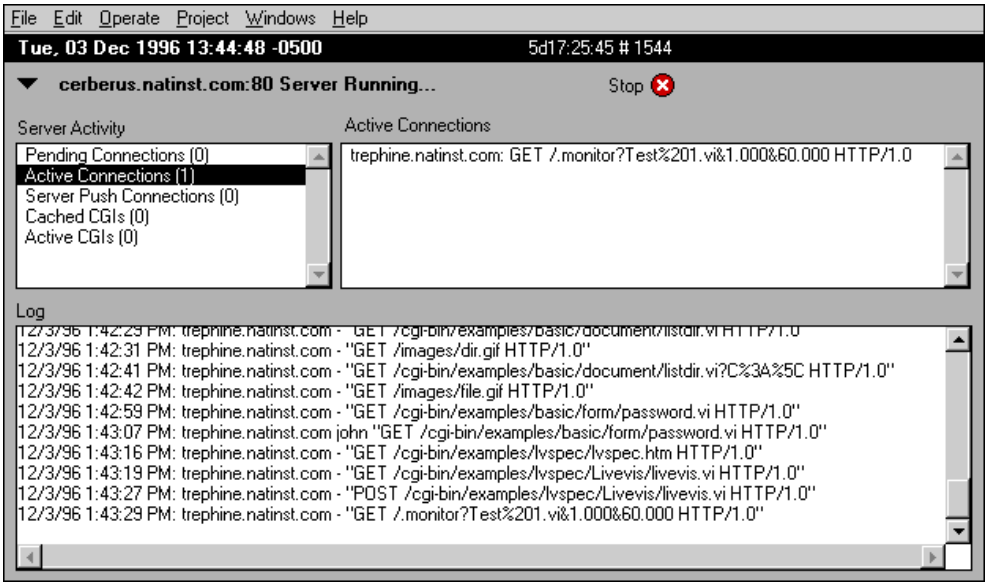

Figure 7-2. HTTP Server in Detailed Mode

In detailed mode, the front panel of the HTTP Server displays the same information as in simple mode plus the following information:

- **Server Activity**—Displays information about different server aspects. When you select an item in the list, the box to the right displays more detailed information. You can choose from the following list of items:
  - **Pending Connections**—Indicates the number of connections the server has accepted but not yet processed. Select this item to display the addresses of these connections.
  - **Active Connections**—Indicates the number of all open connections. Select this item to display the addresses and requests of these connections.
  - **Server Push Connections**—Indicates the number of current server-push, or image animation, connections. Select this item to display addresses and requests of all server-push connections.

  - **Cached CGIs**—Indicates the number of CGI VIs currently in memory. Select this item to list the CGI names, their activity status, and the number of pending requests for each CGI.
  - **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 consist of the date, time, remote system address, username or -, and the request as sent by the client. Error lines depend on the error condition, such as an invalid Universal Resource Locator (URL). You can find more complete log information in the server log files.

How Do I Configure My G Web Server?

You can configure the most common settings for the G Web Server by selecting **Project»Internet Toolkit»Internet Toolkit Configuration…** and choosing **HTTP (Web Server)** from the drop-down menu in the **Internet Toolkit Configuration** dialog box, as shown in Figure 7-3.

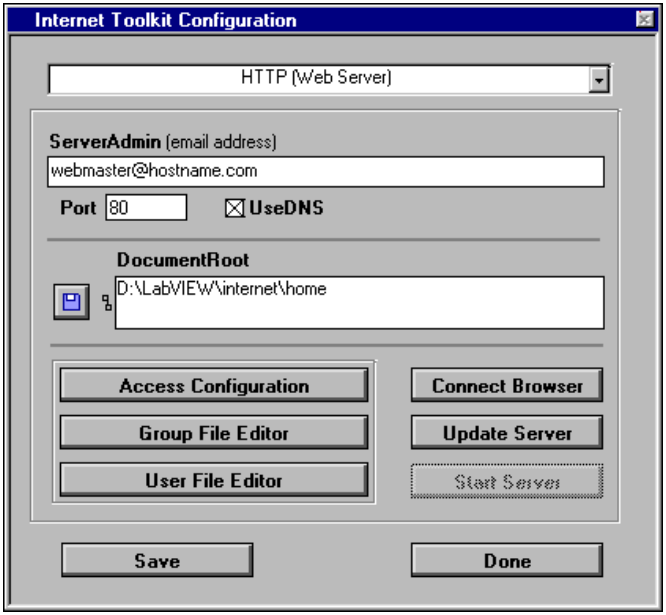

Figure 7-3. Internet Toolkit Configuration Dialog Box

In the **ServerAdmin** field, enter your e-mail address. If the server encounters an error while retrieving a document, it generates a document

with this address to allow viewers of your pages to alert you about this problem.

In the **Port** field, specify the Transmission Control Protocol/Internet Protocol (TCP/IP) port the server is using. The default port for HTTP is port 80. You might have to specify a different port if another HTTP server is already using port 80 on your machine or if you are on a system where you do not have permission to use reserved ports. If you use a non-default port, such as 8000, you have to specify it on URLs that refer to your server, for example `http://hostname:8000/index.htm`.

Deactivate the **UseDNS** checkbox if you do not have access to a DNS server. When UseDNS is enabled, the server converts TCP/IP addresses, such as `130.164.140.14`, to their corresponding host name, such as `www.natinst.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. Deactivate the checkbox if you are uncertain whether you have access to a DNS server.

The **DocumentRoot** field contains the path to the directory in which your HTML documents are located. Use the browse button next to the field to specify the document root path. If you do not specify a valid path, the server tries to use the example home directory that was installed with the Internet Toolkit in *`LabVIEW directory`*`\internet\home`.

Refer to the *Advanced Configuration Options* section in Chapter 8, *Using the G Web Server*, for more information on configuration options.

What Is HTML?

HTML is a markup language for hypertext used in World Wide Web clients. HTML documents consist of plain text with embedded *tags*. You use tags for sectioning HTML documents, formatting the text, setting styles and colors, embedding images and objects in the text, and specifying links to other documents.

Each tag has a unique name enclosed in brackets (`<>`). Most tags also have a closing tag (`</>`). For example, if you want to display the following text: This is a **bold** word.

You use the following string, using the `<B>` (bold) tag for the opening tag and `</B>` for the closing tag:

```
This is a <B>bold</B> word.
```

Some HTML tags use optional additional attributes. For example, the paragraph tag `<P>` can use the `ALIGNMENT` attribute to specify the horizontal alignment of the paragraph. If you do not specify this attribute, the alignment of the paragraph does not change.

```
<P ALIGN=CENTER>This paragraph is centered</P>
```

If plain text in an HTML document contains characters reserved by HTML, such as `<`, `>`, `"`, and `&`, you must encode it according to HTML conventions. For example, the string `3<4` would be written as `3<4` in HTML.

The HTML specification continually changes. Because of this, some companies define specific tags understood only by their browser application but not part of the official standard. This practice usually does not cause problems because the HTML specification states that browsers should ignore tags they do not understand. However, documents that use software-specific tags appear incorrectly in other browsers.

The World Wide Web Consortium (W3C), founded to develop common standards for the World Wide Web, keeps track of versions of HTML. For more information about the state of HTML, refer to the W3C homepage, located at `http://www.w3.org`.

You can design HTML documents using an HTML editor or, if you know the syntax of the language, directly in any text editor. The Internet Developers Toolkit for G provides a library of HTML VIs you can use to construct HTML documents programmatically. To use these VIs effectively, you should be familiar with HTML and the different tags it uses. Several HTML tutorials are available on the Internet, and most bookstores carry a wide selection of books on HTML. Refer to the *Related Documentation* section in the *About This Manual* chapter for more information on HTML documents.

How Do I Get Started with the G Web Server?

One of the most popular features of the Internet Developers Toolkit for G is the ability to view VI front panels remotely using a Web browser. You can perform a simple test of this feature without leaving your computer by selecting **Project»Internet Toolkit»Start HTTP Server…** from a VI front panel.

The TCP/IP host name of your computer appears on the front panel of the HTTP Server window that appears. The name begins after the word Server and ends just before the :80 Running… text, as shown in Figure 7-4. This step assumes you have your computer configured to use the TCP/IP

network protocol. To configure your computer to use the TCP/IP protocol, refer to the documentation for your operating system. Port 80 is the default port all HTTP servers use.

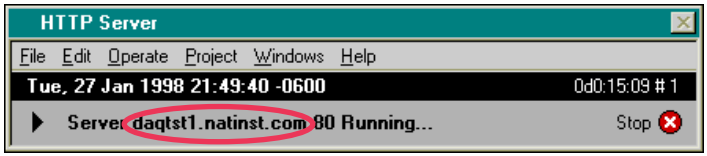

Figure 7-4. Example of TCP/IP Host Name

Open a Web browser so you can experiment with the G Web Server. Microsoft Internet Explorer does not support the server-push animation method this toolkit uses for continuous animation and therefore does not allow you to view animated VI front panels using the `.monitor` URL. You can still view static images of VI front panels, however. Refer to the *Animated Front Panel Image (.monitor URL)* section in Chapter 8, *Using the G Web Server*, for more information.

Type the following string into the URL field of your Web browser or select **File»Open** and type the string:

hostname/`.snap?HTTP+Server`

Replace *hostname* with the name of your computer, as displayed on the HTTP Server VI front panel. If there was no name displayed or this technique does not work, substitute the word `localhost` or `127.0.0.1` for *hostname*.

In this case, the plus sign (+) replaces the whitespace in the name of the VI because a URL cannot have spaces. You also can replace special characters with their *hexadecimal* value preceded by a percent (%) sign. In the case of the whitespace, you would use `%20` 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. As with all URLs, case is unimportant, but any filename extension is crucial. Unlike the HTTP Server VI, most of your VIs might have a `.vi` extension, which you must include as part of the URL. Every time you retype this URL or click the **Reload** or **Refresh** button on your browser, you should see a new image of the specified VI front panel.

Note ***You can quickly build HTML documents with embedded front panel images by selecting* Project»Internet Toolkit»HTML Document Builder….**

Online Examples

The best way to learn about the capabilities of the G Web Server is to examine the online examples provided with the toolkit. To access these examples, use your Web browser to view the default example homepage provided with the toolkit. This section assumes that your server configuration specifies the default document root directory as `LabVIEW directory\internet\home`. Refer to the *How Do I Configure My Toolkit?* section in Chapter 2, *Configuration*, for more information.

To access the default example homepage, type your computer host name, `localhost`, or `127.0.0.1` in the URL field of your Web browser. You should see a G Web Server graphic followed by a **CGI Examples** link. If you see something else, your document root directory is not properly configured. Refer to the *How Do I Configure My Toolkit?* section in Chapter 2, *Configuration*, for more information.

If you want to see examples of some of the things you can do with the G Web Server, follow the **CGI Examples** link. Nearly all the features of the G Web Server are demonstrated, such as static and animated panel images, password protection, form processing via CGI VIs, and *cookies*.

To understand how the examples were created, you should be familiar with HTML code. Refer to the *What Is HTML?* section earlier in this chapter for more information. To examine the 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 of the built-in examples invoke CGI VIs, which is apparent if the `HREF` for the link contains the name of a VI. You can examine these CGI VIs by opening them from within LabVIEW or BridgeVIEW.

CGI Basics

Simply put, a CGI routine is any program capable of accepting information from a client, which is a Web browser, processing that information in some way, and sending a response back to the client, usually in the form of HTML code. For example, most Web pages that contain HTML form elements, such as ring menus, text fields, and radio buttons, use CGI routines to process the choices a user makes and to generate a response. You can use the CGI capabilities of the Internet Developers Toolkit for G to implement similar behavior.

With the Internet Toolkit, you create CGI VIs using standard G programming techniques. To see examples of CGI VIs in action, visit the

default example homepage. Refer to the preceding *Online Examples* section for more information on how to bring up the default server homepage. Follow the **CGI Examples»CGI Basics** links. In particular, examine the **CGI Call with Multiple Parameters (POST)** example.

As explained on that example page, clicking the **Submit** button invokes the CGI VI `/cgi-bin/examples/post_mlt.vi`. You can observe how a CGI VI is associated with a link or a **Submit** button by examining the HTML code for the example page. This particular CGI reads the values of the form elements as selected by the user and returns a table that summarizes the choices. Figure 7-5 shows the block diagram of the `post_mlt` VI.

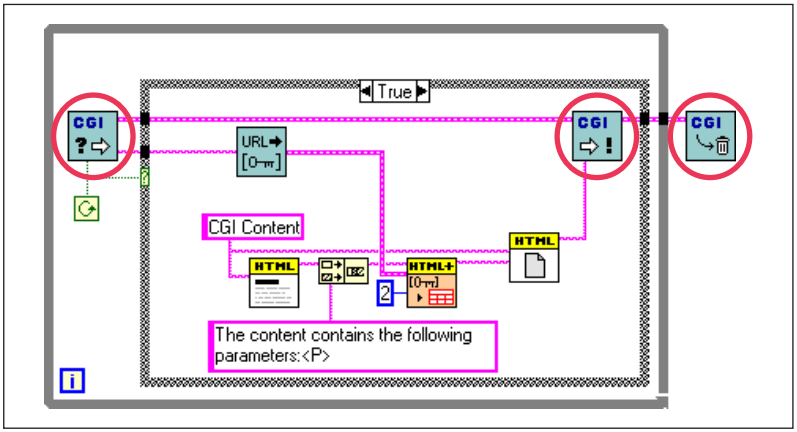

Figure 7-5. post_mlt VI Block Diagram

The circled *subVIs* in Figure 7-5 are crucial subVIs that every CGI VI you create should contain. The leftmost subVI is responsible for obtaining the browser environment and any content string the client submits and for providing that information to the rest of the block diagram in the form of a *Keyed Array* and a string. The next circled subVI to the right is responsible for sending a response back to the client to display in the browser. Typically, this response comes in the form of regular HTML or a reference to an existing URL. The last circled subVI frees the resources associated with this particular CGI call.

The VIs not circled in Figure 7-5 are specific to decoding the form elements that were passed to the CGI in this example using the `POST` method and also are specific to building an HTML table to return to the client browser. Refer to Chapter 8, *Using the G Web Server*, for more information on the CGI and HTML VIs.

8

Using the G Web Server

This chapter explains how to work with the G Web Server.

How Do My VIs Work with the G Web Server?

You can use the G Web Server to passively monitor the front panels of running VIs, to actively execute CGI VIs on behalf of a remote user request, or embed images in HTML documents.

View Running VIs

You can use the G Web Server to publish the front panel image of any VI in memory. To view the static or animated image of a front panel, an HTML page must contain an embedded image tag whose source URL references the G Web Server and specifies the VI name and, optionally, the image parameters.

For static images, the URL takes the following form:

```
<http://web.server.addr/.snap?VI_Name>
```

The `?` character separates the URL from the parameters. `VI_Name` specifies the name of the returned VI front panel image. You must encode the `VI_Name` according to URL naming rules. Replace special characters with their hexadecimal value preceded by a percent (%) sign and replace spaces with a plus (+) sign. Following the VI name, you can add parameters that specify the image format to use (JPEG or PNG), the pixel depth to return (1, 4, 8, or 24 bits), and any additional image parameters, such as image quality for JPEG and compression level for PNG. Refer to the *Front Panel Image Formats* section later in this chapter for more information.

For example, you can write the URL for the static image of the VI `Test Example.vi` in PNG format and in black and white as follows:

```
<http://web.server.addr/.snap?Test+Example.vi
&type=png&depth=1>
```

For animated images, the URL takes the following form:

```
<http://web.server.addr/.monitor?VI_Name>
```

The G Web Server uses the server-push method to implement animations of front panel images. During a server push, the server maintains an open connection and sends a new image after a predefined period of time. Because only a few browsers currently work with server-push animation, browsers that do not work with it receive a single static image instead. You can use all the optional parameters for static images with animated images. You also can specify the refresh rate and the length of an animation.

For example, you can write the URL for the animated image of the VI `Test Example.vi`, which updates once every 2 seconds for 3 minutes, as follows:

```
<http://web.server.addr/.monitor?Test+Example.vi
&refresh=2&lifespan=180>
```

Execute VIs

You can use the G Web Server to load and execute specially designed VIs dynamically. For the server to recognize a VI as a CGI application, you must designate the directory the VI resides in as a CGI directory using the **ScriptAlias** directive. The server uses the CGI to exchange parameters and data with these CGI VIs. You use CGI applications dynamically to create documents whose content frequently changes and to process queries and form requests. Examples of CGI applications include stock market information, fill-out registration forms, and online stores. In the G environment, you can use a CGI application to start or change parameters of an experiment written in G. CGI applications execute on the server system and are different from *applets* you download and execute on the system on which the browser is running.

You can invoke a CGI application from a browser through a link or by clicking the **Submit** button of an HTML form. The URL for a CGI application appears the same as other documents except you often specify additional parameters. The server determines whether a URL describes a regular document or a CGI application.

For example, you can write the URL to execute the CGI VI `params.vi` in the `cgi-bin` directory that includes the two parameters **name** and **age** with values `Bob` and `32`, respectively, as follows:

```
<http://web.server.addr/cgi-bin/params.vi?name=
Bob&age=32>
```

You also can create server-push animations on the G Web Server using CGI VIs. You create a special URL instructing the server to execute the CGI server-push animation. The server then repeatedly calls the CGI when a new image frame is needed. In server-push animations, the CGI must return images instead of HTML documents. You can use the **refresh** and **lifespan** parameters in the same way as in front panel animations.

You can write a URL for a CGI animation using the CGI VI `animate.vi` in the `cgi-bin` directory that includes the parameter **name** with the value `Bob`, updating once every 5 seconds for 1 minute, as follows:

```
<http://web.server.addr/.spool?/cgi-bin/animate.vi
&name=Bob&refresh=5&lifespan=60>
```

Embed Images

You must use the Image (`IMG`) tag and specify the location of the image in the source attribute to embed an image in an HTML document. If the image and the document that contains the image are on the same server, you can use the relative path to the image. If the image and the document are on different servers, you must specify the entire URL.

For example, to embed the image `http://foo/images/sample.gif` in a document on the computer `foo`, use the following HTML code:

```
<IMG SRC="/images/sample.gif">
```

To embed the same image in a document that is not on the computer `foo`, use the following HTML code:

```
<IMG SRC="http://foo/images/sample.gif">
```

The following HTML code describes two headings and two embedded images, a static picture of the VI `test.vi` and an animation of the same VI:

```
<H2>Static Panel Image</H2>
<IMG SRC="/.snap?test.vi" >
<H2>Animated Panel Image</H2>
<IMG SRC="/.monitor?test.vi">
```

Note ***You can quickly build HTML documents with embedded front panel images by selecting*** **Project»Internet Toolkit»HTML Document Builder….**

What URLs Can I Use with My Front Panel Images?

With the G Web Server, you can publish images of your VI front panels on the World Wide Web. You do not need to modify the VIs to display their front panels.

Front Panel Image Formats

The G Web Server can generate images of VI front panels in the *Joint Photographic Experts Group (JPEG)* and *Portable Network Graphics (PNG)* image formats.

The JPEG image format is a public domain image format that all current browsers support. It has been developed for the distribution of real-life images and photographs and uses a lossy compression algorithm to reduce the memory size of an image. When you use JPEG on images that contain lines and text, such as front panels, the resulting image often displays artifacts of the compression, such as fuzzy text or stray color pixels.

The PNG format is a recent public domain image format. The compression algorithm in this format is lossless, which produces PNG images exactly like the original images. PNG is designed to be the successor of the *Graphics Interchange Format (GIF)* format, which also uses lossless compression. PNG is an open standard that you also can use on true-color images. Internet Explorer 4.01 and Netscape Navigator 4.04 support the PNG format. Other browsers require a plug-in or an external application to view PNG images.

Static Front Panel Image (.snap URL)

The `.snap` URL signals the server to return a static image of the front panel of a VI currently in memory. The query parameters in the URL specify the VI name and the attributes of the image.

You must open the front panel of the VI to take snapshots for static images because closed front panels do not update the images of controls when the value changes.

Syntax

```
.snap?VI_Name
    [&type=type]
    [&depth=depth]
    [&quality=quality]
    [&compression=compression]
    [&refresh=refresh]
    [&full=full]
```

`VI_Name`—The name of the returned VI front panel. You must encode the VI name according to HTTP conventions. Replace special characters with `%xx`, where *xx* is the hexadecimal value of the character. You can use the CGI Escape HTTP Param VI to encode the VI name.

`type`—The returned image type, either JPEG or PNG. If no `type` is specified, the default type is used. Refer to the *PanelImageType* directive in Appendix A, *Configuration Directives*, for more information.

`depth`—The depth of the returned image. `depth` can be 1, 4, 8, or 24 bits. If no `depth` is specified, the default depth is used. Refer to the *PanelImageDepth* directive in Appendix A, *Configuration Directives*, for more information.

`quality`—The image quality and memory size of the JPEG front panel image. `quality` can be between 0 and 100. If no `quality` is specified, the default quality is used. Refer to the *PanelImageQuality* directive in Appendix A, *Configuration Directives*, for more information.

`compression`—The compression level used for compressing PNG images. `compression` can be between 0 and 7. If no `compression` is specified, the default PNG compression is used.

`refresh`—The maximum age of a cached image. If a cached image is older than `refresh` seconds, a new image is generated. If no `refresh` is specified, the value of the **PanelImageRefresh** server configuration directive is used instead.

`full`—Specifies whether to return the image of all controls or just the part visible in the window. Set `full` to `on` to indicate all controls and `off` to indicate the window content. If no `full` is specified, the image of the visible front panel in the window is returned.

Examples

- To return the front panel image of the VI `My VI.vi` from the computer `foo` using the default image `type`, `depth`, and `quality`, use the following code:

  ```
  <http://foo/.snap?My%20VI.vi>
  ```

- To return the front panel image of the VI `Test 1.vi` from the computer `foo` using image `depth = 24` and image `type = PNG`, use the following code:

  ```
  <http://foo/.snap?Test%201.vi&depth=24&type=png>
  ```

- To embed the image of the VI `Example.vi` running on the same system as the page that contains this HTML tag, use the following code:

  ```
  <IMG SRC="/.snap?Example.vi">
  ```

- To embed the image of the VI `Example.vi` running on the computer `foo`, use the following code:

  ```
  <IMG SRC="http://foo/.snap?Example.vi">
  ```

Animated Front Panel Image (.monitor URL)

The `.monitor` URL signals the server to return an animated image of the front panel of a VI currently in memory. The query parameters in the URL specify the VI name, attributes of the animation, and attributes of the image. The server accomplishes this animation by taking subsequent snapshots of the front panel image and sending them to the client. Only clients that work with server-push animations, such as Netscape Navigator, receive the animated image. Other browsers receive one static image.

You must open the front panel of the VI to take snapshots for animated images because closed front panels do not update the images of controls when the value changes.

Syntax

```
.monitor?VI_Name
    [&refresh=refresh]
    [&lifespan=lifespan]
    [&type=type]
    [&depth=depth]
    [&quality=quality]
    [&compression=compression]
    [&full=full]
```

`VI_Name`—The name of the returned VI front panel. You must encode the VI name according to HTTP conventions. Replace special characters with `%xx`, where `xx` is the hexadecimal value of the character. You can use the CGI Escape HTTP Param VI to encode the VI name.

`refresh`—The number of seconds after which a new snapshot is sent. If no `refresh` is specified, the default refresh rate is used. Refer to the *ServerPushRefresh* directive in Appendix A, *Configuration Directives*, for more information.

`lifespan`—The number of seconds the front panel animation lasts. `lifespan=0` implies that the animation continues until the browser cancels it. If no `lifespan` is specified, the default lifespan is used. Refer to the *ServerPushLifespan* directive in Appendix A, *Configuration Directives*, for more information.

`type`—The returned image type, either JPEG or PNG. If no `type` is specified, the default type is used. Refer to the *PanelImageType* directive in Appendix A, *Configuration Directives*, for more information.

`depth`—The depth of the returned image. `depth` can be 1, 4, 8, or 24 bits. If no `depth` is specified, the default depth is used. Refer to the *PanelImageDepth* directive in Appendix A, *Configuration Directives*, for more information.

`quality`—The image quality and memory size of the JPEG front panel image. `quality` can be between 0 and 100. If no `quality` is specified, the default quality is used. Refer to the *PanelImageQuality* directive in Appendix A, *Configuration Directives*, for more information.

`compression`—The compression level used for compressing PNG images. `compression` can be between 0 and 7. If no `compression` is specified, the default PNG compression is used.

`full`—Specifies whether to return the image of all controls or just the part visible in the window. Set `full` to `on` to indicate all controls and `off` to indicate the window content. If no `full` is specified, the image of the visible front panel in the window is returned.

Examples

- To generate an animated front panel image of the VI `My VI.vi` from the computer `foo`, using the default `refresh`, `lifespan`, image `type`, `depth`, and `quality`, use the following code:

  ```
  <http://foo/.monitor?My%20VI.vi>
  ```

- To generate a 60-second animation of the front panel image of the VI `Test 1.vi` from the computer `foo` using the default image `type` and `quality` but using `refresh=5`, use the following code:

  ```
  <http://foo/.monitor?Test%201.vi
  &refresh=5&lifespan=60>
  ```

- To embed an animation of the VI `Example.vi` running on the same system as the page that contains this HTML tag, use the following code:

  ```
  <IMG SRC="/.monitor?Example.vi">
  ```

- To embed an animation of the VI `Example.vi` running on the computer `foo`, use the following code:

  ```
  <IMG SRC="http://foo/.monitor?Example.vi
  ```

Animated Front Panel Image (.spool URL)

The `.spool` URL signals the server to return animated front panel images a CGI VI produces. The query parameters in the URL specify the CGI, attributes of the animation, and the CGI parameters. The server accomplishes this animation by repeatedly calling the CGI and returning its output as a server-push animation. Only clients that work with server-push animations, such as Netscape Navigator, receive the animated image. Other browsers receive one static image.

Syntax

```
.spool?CGI_path
    [&refresh=refresh]
    [&lifespan=lifespan]
    [&params]
```

`CGI_path`—The path to the CGI. You can make the path relative to the location of the document that embeds the `.spool` image tag.

`refresh`—The number of seconds after which a new snapshot is sent. If no `refresh` is specified, the default refresh rate is used. Refer to the *ServerPushRefresh* directive in Appendix A, *Configuration Directives*, for more information.

`lifespan`—The number of seconds the front panel animation lasts. `lifespan=0` implies that the animation continues until the browser cancels it. If no `lifespan` is specified, the default lifespan is used. Refer to the *ServerPushLifespan* directive in Appendix A, *Configuration Directives*, for more information.

`params`—A list of parameters sent to the CGI, separated by an ampersand (`&`). If your CGI returns images of a VI front panel, you can specify that `CGI Get Panel Image.vi` interpret the same image parameter that appears in the `.snap` URL.

Examples

- To generate an animation of the documents the CGI VI `/cgi-bin/movie.vi` returns using the default `refresh` and `lifespan`, use the following code:

  ```
  <http://foo/.spool?/cgi-bin/movie.vi>
  ```

- To generate an animation of the documents the CGI VI `/cgi-bin/data.vi` returns using the default `refresh` and `lifespan`, use the following code:

  ```
  <http://foo/.spool?/cgi-bin/data.vi
  &base=2.3&offset=1000>
  ```

  The CGI receives the parameters **base** and **offset** with the values `2.3` and `1000`, respectively.

- To generate a 60-second animation of the documents the CGI VI `/cgi-bin/movie.vi` returns from the computer `foo`, use the following code:

  ```
  <http://foo/.spool?/cgi-bin/movie.vi&refresh=5
  &lifespan=60&name=John%20Lennon>
  ```

  The CGI receives the parameter **name** with the value `John Lennon` and `refresh=5`.

- To embed an animation of the documents the CGI VI `test.vi` returns from the same directory as the page that contains this HTML tag, use the following code:

  ```
  <IMG SRC="/.spool?test.vi">
  ```

- To embed an animation of the documents the CGI VI `/cgi-bin/movie.vi` returns from the same system as the page that contains this HTML tag, use the following code:

  ```
  <IMG SRC="/.spool?/cgi-bin/movie.vi">
  ```

- To embed an animation of the documents the CGI VI `/cgi-bin/movie.vi` returns from the computer `foo`, use the following code:

  ```
  <IMG SRC="http://foo/.spool?/cgi-bin/movie.vi">
  ```

What Is the Common Gateway Interface?

CGI is a standard for external gateway programs to communicate with information servers such as HTTP servers. The current version is CGI/1.1. You can find additional documentation and specifications for the CGI on the National Center for Supercomputing Applications Web site, located at `http://hoohoo.ncsa.uiuc.edu/cgi`.

Note ***The term CGI often is used to describe a CGI application and the interface to that application.***

On the World Wide 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 it. Figure 8-1 shows how this process works.

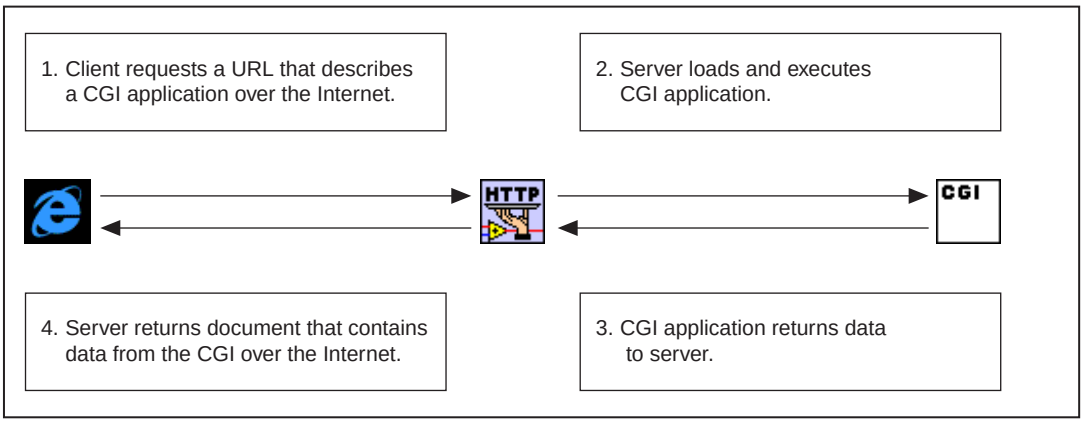

Figure 8-1. CGI Application Process

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

You can supply parameters for your CGI applications in the following two ways, depending on the method with which a CGI is invoked:

- `POST` method—Used only with HTML forms when the user clicks a **Submit** button.
- `GET` method—Used mainly in HTML links but also in HTML forms.

When the user fills out your HTML `POST` form and clicks the **Submit** button, the browser encodes the contents of the fields into an ampersand (`&`)-separated list of `name=value` parameter pairs and sends this string as the content of the `POST` request. If your form contains the fields `name` and `age` and the user enters `John Smith` and `27`, respectively, the browser sends the 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.

With the `GET` method, you use a question mark to separate the CGI parameter from its name. When your form uses the `GET` method to invoke a CGI application, the encoded string is added to the end of the CGI name. Most often, though, an HTML link with the parameters explicitly specified invokes a CGI through the `GET` method. For example, the following URL connects to the server `some.server.adr`, invokes the CGI `/cgi-bin/example.vi`, and passes the parameter **single parameter**:

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

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

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

CGI VIs with the HTTP Server

According to the CGI specification, a server and a CGI application communicate through *environmental variables* and through standard inputs and outputs. Specifically, when the server executes a CGI application, it passes it information in the environmental variables, shown in Table 8-1. The CGI application also can read data from standard input and write any data it generates to standard output.

Table 8-1. CGI Environment Variables

Name	Description
GATEWAY_INTERFACE	Version of the interface, currently CGI/1.1.
SERVER_SOFTWARE	Name and version of the G Web Server.
SERVER_NAME	Name of the computer running the G Web Server as configured or determined by the server.
SERVER_PORT	TCP port at which the server listens for requests.
DOCUMENT_ROOT	Root directory, in UNIX format, of your server documents.
REMOTE_HOST	Domain name or IP address of the remote system the client uses to connect.
REMOTE_ADDR	IP address of the remote system the client connects from.
SCRIPT_NAME	Virtual path to the CGI VI.
REQUEST_METHOD	Method by which you invoke the CGI, either GET or POST.
SERVER_PROTOCOL	Protocol over which the client communicates with the server, currently HTTP/1.0.
HTTP_REFERER	URL of document that contains the link that invoked this CGI.
HTTP_USER_AGENT	Browser software the remote client uses.
HTTP_ACCEPT	List of MIME-like types that the browser understands.
QUERY_STRING	URL-encoded parameters sent to this CGI.
REMOTE_USER	Username of client.
REMOTE_IDENT	User password of client.

Because the G environment does not work with standard input and output, a CGI VI receives the standard input data as a string when it receives a

request. Similarly, instead of writing to standard output, the CGI VI passes any data it generates as a string.

Figure 8-2 shows the basic structure of a CGI application.

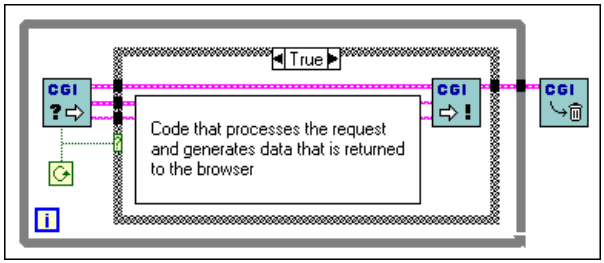

Figure 8-2. Structure of a CGI Application

The CGI application in Figure 8-2 consists of the following three VIs. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»CGI VIs»CGI Template VIs**. For more information about these VIs, right-click on the VI in the block diagram and select **Online Help** from the VI pop-up menu.

- CGI Read Request VI—Waits for and reads an HTTP request for the CGI VI.
- CGI Write Reply VI—Sends the HTTP reply to the browser.
- CGI Release VI—Informs the server that the CGI application finished executing and that it can be released from memory.

What Are CGI Utility VIs?

Many CGI applications execute common tasks, such as looking at the `QUERY_STRING` environmental variable to perform their task. Because of this, the G Web Server contains VIs that implement these common functions. These CGI Utility VIs are grouped in the following areas:

- CGI Parameter VIs—Convert to and from URL-encoded parameters.
- File Path VIs—Convert between platform-specific and virtual or UNIX-format paths.
- Network VIs—Provide network support.
- Panel Image VIs—Produce images from other VI front panels.
- Keyed Array VIs—Build and index data in Keyed Array format.

- HTML VIs—Generate HTML code.
- State Information VIs—Maintain information between multiple CGI invocations.

Note ***For more information on the State Information VIs, refer to the*** *How Do I Maintain Client-State Information?* ***section later in this chapter.***

CGI Parameter VIs

URLs do not permit some characters as part of their code, such as spaces or ASCII characters with values greater than 127. Other characters, such as `?`, `&`, `/`, `:`, `;`, and `=` have special meaning when used with HTTP. Whenever you use these characters as part of a URL, you must encode them. You can use the CGI Parameter VIs to complete this encoding for you. You replace each special character with a percent (`%`) character followed by two hexadecimal digits that describe its numeric value. You also can replace the space character with the plus (`+`) character. For example, to pass the string `War & Peace` to the CGI VI `test.vi`, the URL contains the following code:

```
test.vi?War%20%26%20Peace
```

When you pass several parameters to a CGI VI, you pass them as an ampersand (`&`)-separated list of `name=value` pairs. For example, to pass the parameters **name** and **age** with values `John Smith` and `27`, respectively, to the CGI VI `test.vi`, the URL contains the following code:

```
test.vi?name=John%20Smith&age=27
```

You can use the following VIs to convert between URL-encoded strings and normal data. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»CGI VIs**. For more information about these VIs, right-click on the VI in the block diagram and select **Online Help** from the VI pop-up menu.

- CGI Escape HTTP Parameter VI—Encodes a string according to URL conventions.
- CGI Unescape HTTP Parameter VI—Decodes a URL-encoded string.
- CGI Parse URL-Encoded Param String VI—Converts a URL-encoded parameter string into a Keyed Array.
- CGI Build URL-Encoded Param String VI—Converts a Keyed Array into a URL-encoded parameter string.
- CGI Get Query Parameters VI—Converts the `QUERY_STRING` environmental variable into a plain-text string and parameter list.

File Path VIs

URLs consist of a method part, host, and a virtual path to a document, in UNIX format. For example, the URL `http://www.natinst.com/labview/internet` uses the `http` method, the host `www.natinst.com`, and the virtual path `/labview/internet`. A *virtual path* is a path the server can use to redirect the request to another URL, alias it into another path, or add it to the end of the path of the document root directory.
The File Path VIs convert your platform-specific paths to virtual and UNIX-style paths. For example, if you use Windows to run the server `www.natinst.com` with the document root directory `c:\www`, the virtual path `/labview/internet` refers to the absolute path `c:\www\labview\internet`.

You can use the following VIs to convert between platform-specific paths and virtual and UNIX-style paths. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»CGI VIs**. For more information about these VIs, right-click on the VI in the block diagram and select **Online Help** from the VI pop-up menu.

- CGI Get Info VI—Returns the CGI VI name, absolute path, and virtual path through which you invoked the application.
- CGI Unix to Path VI—Converts a string that contains a UNIX-style path to a G path.
- CGI Path To Unix VI—Converts a G path to a string that contains a UNIX-style path.
- CGI Build Unix Path VI—Builds a UNIX-style path from a base path and a second path. If the second path is relative, it is added to the end of the base path. If it is absolute, it is used instead of the base path.
- CGI Translate Virtual Path VI—Translates a virtual path to an absolute path.
- CGI Script Relative Path VI—Returns a path relative to the CGI application. This is useful when your CGI application stores data files in its own directory.

Network VIs

CGI applications can use the regular Transmission Control Protocol (TCP) VIs to convert an IP address to a domain name or vice versa. However, the following Network VI might work better. This VI looks at the server **UseDNS** configuration directive setting and does not try to obtain a domain name when the *Domain Name System (DNS)* lookup is turned off.

You can find this VI in the **Functions** palette by selecting **Internet Toolkit»CGI VIs**. For more information about this VI, right-click on the VI in the block diagram and select **Online Help** from the VI pop-up menu.

- CGI IP Name To Name VI—Takes a domain name or IP address and returns its IP address and domain name. It attempts DNS lookup only according to the server configuration. Refer to the *UseDNS* directive in Appendix A, *Configuration Directives*, for more information.

Panel Image VIs

Using the Panel Image VIs, you can return the image of another VI front panel. With the Panel Image VIs, you can specify the name and certain image parameters to capture the VI front panel image. The VIs return the image data and an appropriate header string that correctly identifies the image type.

You can use the following VIs to produce these front panel images. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»CGI VIs**. For more information about these VIs, right-click on the VI in the block diagram and select **Online Help** from the VI pop-up menu.

- CGI Get Panel Image VI—Uses a VI name and a Keyed Array parameter and returns the image of the VI specified by the name with image parameters specified by the Keyed Array.
- CGI Get Panel JPEG Image VI—Uses a VI name and image parameters and returns the front panel image in JPEG format.
- CGI Get Panel PNG Image VI—Uses a VI name and image parameters and returns the front panel image in PNG format.

Keyed Array VIs

The G Web Server supplies the CGI environmental variables, listed in Table 8-1, in the form of a Keyed Array. A Keyed Array is a data structure similar to an array of strings. However, unlike a regular array, you index elements by a string key instead of a numeric index. Keyed Arrays are similar to the Associative Array in the *Perl* language. Many operations on Keyed Arrays allow you to specify whether string matching should be case sensitive. The values for this parameter are `a != A` for case sensitive, `a == A` for case insensitive, and `system` for same case sensitivity as when comparing filenames. For example, you would use case sensitive on UNIX and case insensitive on Windows and MacOS.

You can use the following VIs to index, add, or remove elements and access the keys and values stored in a Keyed Array. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»CGI VIs»Keyed Array VIs**. For more information about these VIs, right-click on the VI in the block diagram and select **Online Help** from the VI pop-up menu.

- Keyed Array Add VI—Inserts a new element into a Keyed Array.
- Keyed Array Index VI—Indexes an element in a Keyed Array.
- Keyed Array Remove VI—Removes an element from a Keyed Array.
- Keyed Array Clear VI—Removes all elements from a Keyed Array.
- Keyed Array Keys VI—Returns an array of keys of all the elements in the Keyed Array.
- Keyed Array Values VI—Returns an array of values of all the elements in the Keyed Array.
- Keyed Array Contents VI—Returns an array of keys and an array of the corresponding values of all the elements in the Keyed Array.
- Keyed Array Index Wildcards VI—Indexes the keys of the elements in the Keyed Array *wildcard expression*s.
- Keyed Array Wildcard Index VI—Indexes the elements of a Keyed Array using wildcard expressions.
- Keyed Array Equal VI—Returns the value TRUE if all the elements in two Keyed Arrays contain the corresponding keys and values.

HTML VIs

The HTML VI library in this toolkit contains VIs that generate all the tags specified in HTML 3.2, with the exception of the obsolete tags `XMP`, `LISTING`, and `PLAINTEXT`. This toolkit contains a VI for building arbitrary HTML tags if you want to generate tags that are not in the HTML 3.2 specification. The library also includes VIs for building tags with attributes specific to the G Web Server, for front panel images, and for converting arrays into HTML tables.

You wire the HTML VIs in sequence so each VI generates one or more HTML tags. Most VIs also have a **label** input so you can insert extra text between the incoming HTML code and the tag the VI generates. VIs that implement tags without a corresponding closing tag are wired in sequence. VIs that contain a closing tag and enclose some text or HTML code take the enclosed content as a parameter. Optional and required tag attributes also are specified as parameters. Some VIs generate more complex attribute strings for certain HTML tags. Text attributes, with the exception of URLs, are filtered so that the HTML reserved characters are encoded properly.

Figure 8-3 constructs the HTML code for the following text:

`These are` **`words`** `with` ***`style`***

This string also includes a horizontal rule and an embedded picture, `example.gif`, with the alternate description, `Taste & Style`.

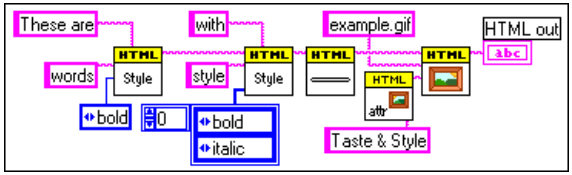

Figure 8-3. HTML Block Diagram

The following is the output of the **HTML out** string:

```
These are<B>words</B>with<I><B>style</B></I><HR>
<IMG SRC="example.gif" ALT="Taste &amp; Style">
```

Some tags make sense only within the content of another tag. For example, an HTML table is defined by the `<TABLE>` tag, which encloses a list of table row tags `<TR>` that in turn enclose table cell tags `<TD>`. Figure 8-4 constructs the HTML code for the following table:

One	Two
1	2

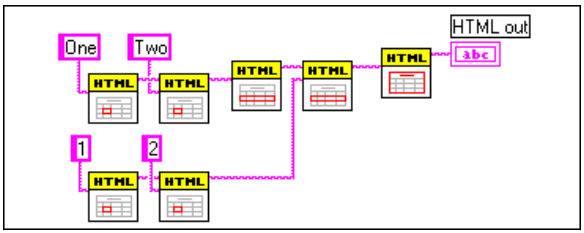

Figure 8-4. HTML Table Block Diagram

The following is the output of the **HTML out** string:

```
<TABLE>
<TR>
<TD>One</TD>
<TD>Two</TD>
</TR>
<TR>
<TD>1</TD>
<TD>2</TD>
</TR>
</TABLE>
```

Because some HTML code, such as building tables out of arrays, is fairly common, this toolkit includes VIs that combine several HTML tags to construct more complicated HTML code. For example, the HTML+ String Array To Table VI constructs a table out of a 2D string array. The block diagram in Figure 8-5 generates the same HTML code shown in Figure 8-4.

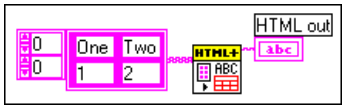

Figure 8-5. HTML+ String Array to Table VI

Some VIs generate tags with parameters or attributes specific to the G Web Server. For example, the `HTML+ Monitor.vi` generates an image tag for a VI front panel animated image. The block diagram in Figure 8-6 constructs HTML code for an animation of `animation example.vi` that refreshes every 5 seconds.

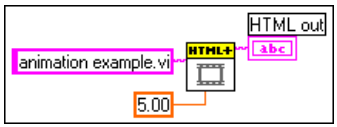

Figure 8-6. HTML+ Monitor VI

The following is the output of the **HTML out** string:

```
<IMG SRC="/.monitor?animation%20example.vi
&refresh=5.000">
```

Using `HTML Generic Tag.vi`, you can create any HTML tag. You also can use many of the CGI Utility VIs to help you construct tag attributes. In

Figure 8-7, the block diagram on the left uses the HTML Font VI to generate a tag for red text. The block diagram on the right constructs the same HTML code using the HTML Color Tag Attribute VI and the HTML Generic Tag VI.

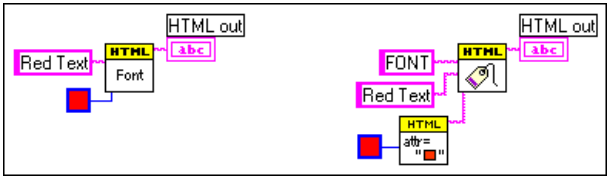

Figure 8-7. HTML Generic Tag VI

The following is the output of the **HTML out** string:

```
<FONT COLOR="#FF0000">Red Text</FONT>
```

The HTML Document VI uses the HTML code you created to build an HTML document. For example, the block diagram in Figure 8-8 builds an HTML document with the title `Acquired Data`, a heading also called `Acquired Data`, a table built from a 2D numeric array with three digits of precision, and a link to the National Instruments Web site called `National Instruments`.

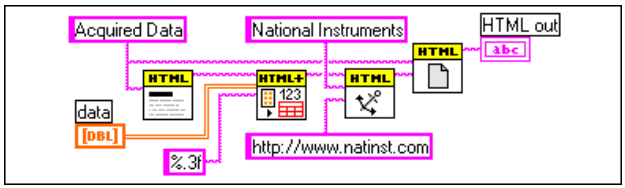

Figure 8-8. HTML Document VI

The following is the output of the **HTML out** string:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2
Draft//EN">
<HTML>
<!-- Constructed with the G Web Server -->
<HEAD>
<TITLE>Acquired Data</TITLE>
</HEAD>
<BODY>
<H1>Acquired Data</H1>
<TABLE>
<TR>
```

```
<TD>1.000</TD>
<TD>2.000</TD>
</TR>
<TR>
<TD>3.000</TD>
<TD>4.000</TD>
</TR>
</TABLE>
<A HREF="http://www.natinst.com">National
Instruments</A>
</BODY>
</HTML>
```

You can use the following VIs to construct your HTML code. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»CGI VIs»HTML VIs**. For more information about these VIs, right-click on the VI in the block diagram and select **Online Help** from the VI pop-up menu.

- HTML Address VI—Constructs the `<ADDRESS>` tag.
- HTML Anchor VI—Constructs the `<A>` tag.
- HTML Applet VI—Constructs the `<APPLET>` tag.
- HTML Area VI—Constructs the imagemap `<AREA>` tag.
- HTML Base VI—Constructs the `<BASE>` tag for the HTML `HEAD` section.
- HTML Basefont VI—Constructs the `<BASEFONT>` tag.
- HTML Blockquote Text VI—Constructs the `<BLOCKQUOTE>` tag.
- HTML Break VI—Constructs the `<BR>` tag.
- HTML Build HREF Parameters VI—Constructs a CGI URL out of a base URL, a query string, and a parameter list.
- HTML Color Tag Attribute VI—Constructs a `<COLOR>` tag attribute from a numeric color value.
- HTML Definition List VI—Constructs a `<DL>` tag with embedded `<DT>` and `<DD>` tags.
- HTML Division VI—Constructs a `<DIV>` tag.
- HTML Document VI—Constructs HTML documents out of HTML code.
- HTML Filter Special Characters VI—Replaces `&`, `"`, `<`, and `>` with `&`, `"`, `<`, and `>`, respectively.
- HTML Font VI—Constructs a `<FONT>` tag.

- HTML Form Button VI—Constructs an HTML form `<INPUT>` tag of type `SUBMIT` or `RESET`.
- HTML Form Control Button VI—Constructs an HTML form `<INPUT>` tag of type `CHECKBOX` or `RADIO`.
- HTML Form File VI—Constructs an HTML form `<INPUT>` tag of type `FILE`.
- HTML Form Hidden Field VI—Constructs an HTML form `<INPUT>` tag of type `HIDDEN`.
- HTML Form Image Button VI—Constructs an HTML form `<INPUT>` tag of type `IMAGE`.
- HTML Form Text Area VI—Constructs an HTML form `<TEXTAREA>` tag.
- HTML Form Text VI—Constructs an HTML form `<INPUT>` tag of type `TEXT` or `PASSWORD`.
- HTML Form VI—Constructs an HTML `<FORM>` tag.
- HTML Generic Tag VI—Constructs a generic HTML tag.
- HTML Heading VI—Constructs an `<H1>`, `<H2>`, `<H3>`, `<H4>`, `<H5>`, or `<H6>` tag.
- HTML Horizontal Rule VI—Constructs an `<HR>` tag.
- HTML Image Attributes VI—Constructs `ISMAP`, `USEMAP`, `ALT`, `ALIGN`, `BORDER`, `HSPACE`, `VSPACE`, `HEIGHT`, and `WIDTH` image tag attributes.
- HTML Image VI—Constructs an `<IMAGE>` tag.
- HTML IsIndex VI—Constructs an `<ISINDEX>` head tag.
- HTML Link VI—Constructs a `<LINK>` head tag.
- HTML List VI—Constructs an `<UL>`, `<OL>`, `<DIR>`, or `<MENU>` tag with embedded `<LI>` tags.
- HTML Map VI—Constructs an image `<MAP>` tag.
- HTML Meta VI—Constructs a `<META>` head tag.
- HTML Numeric Tag Attribute VI—Constructs a numeric tag attribute.
- HTML Numeric Tag Attributes VI—Constructs a list of numeric tag attributes.
- HTML Paragraph VI—Constructs a `<P>` tag.
- HTML Param VI—Constructs an applet `<PARAM>` tag.
- HTML Phrase Style VI—Constructs an `<EM>`, `<STRONG>`, `<DFN>`, `<CODE>`, `<SAMP>`, `<KBD>`, `<VAR>`, or `<CITE>` tag.

- HTML Preformatted Text VI—Constructs a <PRE> tag.
- HTML Table Cell VI—Constructs a table <TD> or <TH> tag.
- HTML Table Row VI—Constructs a table <TR> tag.
- HTML Table VI—Constructs a <TABLE> tag.
- HTML Text Style VI—Constructs <TT>, <I>, <B>, <U>, <STRIKE>, <BIG>, <SMALL>, <SUB>, or <SUP> tags.
- HTML Text Tag Attribute VI—Constructs a text tag attribute.
- HTML Text Tag Attributes VI—Constructs a list of text tag attributes.
- HTML+ Form Hidden Field List VI—Constructs a list of HTML form <INPUT> tags of type HIDDEN.
- HTML+ Form Radio Button Group VI—Constructs a list of HTML form <INPUT> tags of type RADIO.
- HTML+ Form Selection VI—Constructs an HTML form <SELECT> tag with a list of embedded <OPTION> tags.
- HTML+ Keyed Array To Table VI—Constructs a <TABLE> tag with embedded <TR> and <TD> tags from a Keyed Array.
- HTML+ Labeled Table VI—Constructs a <TABLE> tag with embedded <TR>, <TH>, and <TH> tags from a 2D array of strings and arrays of strings describing the row and column headers.
- HTML+Meta List VI—Constructs a list of <META> head tags.
- HTML+ Monitor VI—Constructs an <IMG> tag with a .monitor URL.
- HTML+ Numeric Array To Table VI—Constructs a <TABLE> tag with embedded <TR> and <TD> tags from a 2D array of doubles.
- HTML+ Param List VI—Constructs a CGI URL from a base URL, a query string, and a Keyed Array.
- HTML+ Snapshot VI—Constructs an <IMG> tag with a .snap URL.
- HTML+ Spool VI—Constructs an <IMG> tag with a .spool URL.
- HTML+ String Array To Table VI—Constructs a <TABLE> tag with embedded <TR> and <TD> tags from a 2D array of strings.

How Do I Maintain Client-State Information?

The 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.

Often, it is useful to maintain state information across several connections. For example, imagine you are an online vendor. When a user browses through your online catalog, he or she can add items to a shopping cart. When the user finishes shopping, he or she can choose to purchase the items in the cart. You must maintain the information about the items until the user finishes the purchase. You can store information, such as what items the user chose, in a simple data file. However, this does not work if more than one user is shopping at a time. Because you want to work with multiple users, you must find an alternative information storage method.

You can choose from several approaches to maintain client-state information across multiple accesses. For example, you can insert information you collect into hidden fields of an HTML form, or you could use a *cookie*. A cookie is a token that uniquely identifies some information. In the shopping cart example, the cookie records all the items the user has put into the shopping cart. Using a cookie, you can maintain your information on the client side or on the server side.

Client-Side Cookies

To store state information on the client system, you can use client-side cookies. The advantage of client-side cookies is that they are simple 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 information written to their disk without their knowledge, and potential security risks can arise when client-state information can be sent to a host other than the one that stores the information.

Using Client-Side Cookies

Refer to the previously mentioned online catalog example. Every time a user adds anything to his or her shopping cart, the CGI that deals with the request returns a header line that specifies a cookie with the encoded information. The browser stores this cookie and sends it back to the server with every subsequent request. When the user is ready to purchase the items, the CGI that generates the purchase form reviews all the cookies the browser sends and builds an appropriate invoice.

Client-Side Cookie Example

The first page of your online store might have a form where users fill in their names and then click a button. A CGI application processes this form and returns a document that contains the username in the appropriate places along with a header line that describes a cookie that contains the username. When the browser receives the document, it stores the cookie in its cookie file. Every time the user clicks a link on this page that leads back to the server, the cookie that contains the username is sent along with the request. If another user connects to your store, that person will have a different cookie that contains his or her name stored on their computer.

Client-Side Cookie VIs

The G Web Server provides VIs for client-side cookie management. With these VIs, you can 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 the browser. The following VIs retrieve and set client-side cookies. You can find these VIs in the **Function** palette by selecting **Internet Toolkit»CGI VIs**.

- CGI Get Query Client Side Cookies—Returns the client-side cookies the browser sends, which contain the contents of the `HTTP_COOKIE` entry in the **env** array.
- CGI Set Client Side Cookie—Constructs a `Set-Cookie:` reply header line that specifies a client-side cookie to store. Cookie name and contents are URL encoded.
- CGI Set Multiple Client Side Cookies—Constructs `Set-Cookie:` reply header lines that specify multiple client-side cookies to store. Cookie names and contents are URL encoded.

Server-Side Cookies

To store state information on the server system, you can use server-side cookies. There are several advantages to using server-side cookies:

- No dependence on the browser software
- No data stored on the client system
- Less transmitted data
- Security maintained by the server

However, the disadvantage is that server state information is maintained only for a certain time, after which it automatically expires. This time is specified by the server, not the client.

Using Server-Side Cookies

Refer to the previously mentioned online catalog example. The user enters your online store by clicking on a link to the main document. A CGI generates the main document by creating a cookie and embedding it in the document. Then, each time a user moves to another page on the server, the cookie is passed along with the request for that page. When the user adds an item to his or her shopping cart, the CGI that deals with that request adds the item to the information this cookie maintains. When the user is ready to purchase the items, the CGI that generates the purchase form reviews all the items associated with the cookie and builds the appropriate invoice.

Server-Side Cookie Example

The first page of your online store might have a form on which users fill in their names and then click a button. A CGI application processes this form and returns a document that contains the username in appropriate places and in which all links are tagged with the cookie the CGI application generated, as shown in the following example:

```
<A HREF="page.vi?cookie=02A34CCD&name=strings>
String Instruments</A><BR>
```

```
<A HREF="page.vi?cookie=02A34CCD&name=brasss>Brass
Instruments</A><BR>
```

```
<A HREF="page.vi?cookie=02A34CCD&name=woods>
Woodwind Instruments</A><BR>
```

The CGI Page VI looks at its parameters to generate a new page in which it embeds the cookie. If another user connects to your store, that person receives the same pages except with a different cookie value.

Server-Side Cookie VIs

The G Web Server provides VIs for server-side cookie management. With these VIs, you can create and destroy cookies and add and query information associated with cookies. Several VIs help you create HTTP connection-based cookies and documents that contain cookies.

The G Web Server defines a server-side cookie as a cluster of two strings, cookie ID and address. The cookie ID string is used to identify a specific cookie and usually is embedded in the HTML document a CGI application creates. The address string ensures that only clients with the same address can access a server-side cookie. This way, cookies created by requests from a specific client can be viewed and modified only by that client.

You can use the following VIs to create, modify, and dispose of server-side cookies. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»CGI VIs»Cookie VIs**. For more information about these VIs, right-click on the VI and select **Online Help** from the VI pop-up menu.

- Cookie Create VI—Creates a unique cookie identifier to an empty cookie.
- Cookie Destroy VI—Disposes of a cookie.
- Cookie Add Entry VI—Adds an entry specified by a key and a value to a cookie.
- Cookie Get Entry VI—Returns the value of the entry specified by a key from a cookie.
- Cookie Get Data VI—Returns a Keyed Array that contains all the entries in a cookie.
- Not A Cookie VI—Returns an invalid cookie.
- Not A Cookie? VI—Tests the validity of a cookie.

The G Web Server also provides several CGI Utility VIs that work with server-side cookies in CGI applications. These VIs identify the cookies as parameters named **Magic_Cookie**.

You can use the following VIs to work with server-side cookies in HTML documents. You can find these VIs in the **Functions** palette by selecting **Internet Toolkit»CGI VIs**. For more information about these VIs, right-click on the VI and select **Online Help** from the VI pop-up menu.

- CGI Cookie VI—Returns the cookie associated with the CGI request or creates a new cookie given the **env** parameter and parameters of a CGI request if a **Magic_Cookie** parameter is present.
- CGI Spool Cookie VI—Returns or creates a cookie associated with a `.spool` CGI request based on the **cgi connection info** parameter the CGI Read Request VI returns.
- CGI Add Params To Cookie VI—Adds parameters to a cookie.
- CGI Build Cookie Document VI—Loads a file from disk and replaces every occurrence of the string `-*Cookie*-` with a cookie ID.

Advanced Configuration Options

When you configure your G Web Server, you edit the server configuration files to specify how the server behaves on your system.

The following rules apply to all the G Web Server configuration files:

- Case-insensitive entries—Configuration entries are not case sensitive, except where pathnames on case-sensitive file systems are involved.
- Comment lines beginning with #—Lines to ignore must begin with #.
- One directive per line—Each line consists of:

  `Directive data [data2 ... data`*`n`*`]`

  - `Directive`—A keyword the server recognizes followed by whitespace. Refer to Appendix A, *Configuration Directives*, for more information on each directive.
  - `data`—Specific to each directive. Additional data entries are separated by whitespace.
- Extra whitespace ignored—Embedded spaces between `Directive` and `data` are ignored. To embed a space in the data without separating it from any subsequent arguments, use a backslash (`\`) character before the space.
- Paths in UNIX format—The path separator is `/`. All absolute paths must begin with this character.
  - On Windows, the following code sets **PathDirective** to the path `C:\labview\web`:
    `PathDirective /c/labview/web`
  - On MacOS, the following code sets **PathDirective** to the path `Macintosh HD:LabVIEW`. Notice that a `\` precedes the space.
    `PathDirective /Macintosh\ HD/LabVIEW`

You must edit the following three main configuration files for your G Web Server:

- Access Configuration—Controls client access to certain directories and determines which features are available in different directories.
- Server Configuration—Controls the technical aspects of server operation.
- Server Resource Map Configuration—Controls document and script locations and aliases.

Access Configuration

The Access Configuration controls the following aspects of document directories in a tree with the server:

- Access—Restricts access to a branch of the directory tree to only hosts and authenticated users you specify.
- Server Features—Secures directories by disabling certain server functions.

You can use the following two files to control access to directories:

- Global Access Configuration File—A document in your server `conf` directory, specified by the Server Configuration directive **AccessConfig**, that controls access to any directory in your tree.
- Per-Directory Access Configuration File—A file, within the document tree and with the name specified by the Server Resource Map Configuration directive **AccessFileName**, that controls access to the file directory and to any subdirectories.

Configuration Directives

In addition to the general configuration rules that apply to all HTTP configuration files for G, the access control files also work with directives and *sectioning directives*.

You can use the following directives and sectioning directives to write an Access Configuration file. Refer to Appendix A, *Configuration Directives*, for more information on each directive.

- **Directory**
- **Panel**
- **AllowOverride**
- **AuthName**
- **AuthType**
- **AuthUserFile**
- **AuthGroupFile**
- **Limit**
  - **Order**
  - **Deny**
  - **Allow**
  - **Require**

  - **Satisfy**
  - **Referer**
  - **OnDeny**

Server Configuration

The Server Configuration controls the operation of the server. Use the Server Configuration file `lvhttp.cfg`, located in the `conf` directory, to complete the configuration.

Configuration Directives

You can use the following directives to configure your Server Configuration file. Refer to Appendix A, *Configuration Directives*, for more information on each directive.

- **Port**
- **UseDNS**
- **ServerAdmin**
- **ServerRoot**
- **ServerName**
- **StartServers**
- **TimeOut**
- **AccessConfig**
- **ResourceConfig**
- **TypesConfig**
- **CGICacheTime**
- Logging directives
  - **ErrorLog**
  - **TransferLog**
  - **AgentLog**
  - **RefererLog**
  - **LogOptions**
- Front panel images directives
  - **PanelImageType**
  - **CheckPNGSupport**
  - **PanelImageDepth**

  - **PanelImageQuality**
  - **PanelImageRefresh**
  - **PanelImageCacheCompactTime**
- Server-push animations directives
  - **ServerPushRefresh**
  - **ServerPushLifespan**
  - **ServerPushMaxConnections**
  - **ServerPushMinTime**

Server Resource Map Configuration

The Server Resource Map Configuration controls the presentation of the server documents. Use the Server Resource Map file `srm.cfg`, located in the `conf` directory, to complete the configuration.

Configuration Directives

You can use the following directives to configure your Server Resource Map file. Refer to Appendix A, *Configuration Directives*, for more information on each directive.

- **DocumentRoot**
- **AccessFileName**
- Virtual URLs directives
  - **Redirect**
  - **Alias**
  - **ScriptAlias**
- **AddType**
- **AddEncoding**
- **DefaultType**
- Indexing options directives
  - **DirectoryIndex**
  - **FancyIndexing**
  - **DefaultIcon**
  - **ReadmeName**
  - **HeaderName**
  - **AddIcon**

  - **AddIconByType**
  - **AddIconByEncoding**
  - **IndexIgnore**
- **ErrorDocument**

A

Configuration Directives

This appendix describes the configuration directives for the G Web Server and the use of wildcard expressions within the directives.

Wildcard Expressions

The G Web Server works with wildcard expressions you can use to specify patterns that match strings. The G Web Server wildcard expressions use formats similar to UNIX shell wildcard patterns. Both use the characters `*` and `?`, where `*` specifies zero or more of any character and `?` specifies exactly one instance of any character. To match the `*` or `?` character, begin the expression with a `\`.

The following list provides some examples of wildcard expressions:

- `foo*` matches `foo`, `fooa`, and `foobar`, but does not match `afoo`.
- `*/.??*` matches any substring that contains `/.` followed by two or more characters. You can use this pattern for matching UNIX hidden files such as `/foo/bar/.htaccess`.
- `*\?` matches any substring that ends in a question mark.

You can use these wildcard expressions in many of the following configuration directives.

Access Configuration Directives

The Access Configuration directives control the following aspects of document directories in a tree with the server:

- Access—Restricts access to a branch of the directory tree to only hosts and authenticated users you specify.
- Server Features—Secures directories by disabling certain server functions.

For more information on the Access Configuration specification, refer to the *Access Configuration* section in Chapter 8, *Using the G Web Server*.

Note ***You also can configure the Global Access Configuration File (ACF) by selecting* Project›Internet Toolkit›Internet Toolkit Configuration… *and choosing* HTTP (Web Server) *from the drop-down menu in the* Internet Toolkit Configuration *dialog box. Click the* Access Configuration *button to display the* HTTP Access Configuration *dialog box.***

Directory

The **Directory** sectioning directive specifies the directory to which access control directives apply.

Scope

This directive applies only to the Global ACF. You must store all directives in the Global ACF in a **Directory** or **Panel** section.

Syntax

Opening directive:

```
<Directory dir>
```

`dir` is the absolute pathname of the directory you are protecting. It also can be a wildcard expression of a set of directories you want to protect.

Closing directive:

```
</Directory>
```

Filename

`access.cfg`

Example

The following directive specifies that the directives contained in the **Directory** section apply only to the server directory `/c/web` and its subdirectories:

```
<Directory /c/web>
allow from *.natinst.com
</Directory>
```

Panel

The **Panel** sectioning directive specifies the virtual instrument (VI) front panel to which access control directives apply.

Scope

This directive applies only to the Global ACF. You must store all directives in the Global ACF in a **Directory** or **Panel** section.

Syntax

Opening directive:

```
<Panel vi>
```

`vi` is the name of the VI you are protecting. It also can be a wildcard expression of a set of VIs you want to protect.

Closing directive:

```
</Panel>
```

Filename

`access.cfg`

Example

The following directive specifies that the directives contained in the **Panel** section apply only to those VIs ending with `.vi`:

```
<Panel *.vi>
deny from all
</Panel>
```

AllowOverride

The **AllowOverride** directive controls which Access Configuration directives you can overrule on a per-directory ACF.

You cannot restrict the Global ACF with this directive.

Scope

This directive appears only in the Global ACF.

Syntax

`AllowOverride or1 or2 ... or`*n*

`or` is one of the following:

- `None`—You cannot use ACFs in this directory.
- `All`—You can use any ACF in this directory.
- `AuthConfig`–You can use the following directives:
  - **AuthName**
  - **AuthType**
  - **AuthUserFile**
  - **AuthGroupFile**
- `Limit`—You can use the **Limit** sectioning directive.

Filename

`access.cfg`

Default

If no **AllowOverride** is specified, the server assumes the following:

```
AllowOverride All
```

Example

The following directive overrides the **Limit** sectioning directive for the ACFs in this directory:

```
AllowOverride Limit
```

AuthName

The **AuthName** directive sets the name of the authorization realm for this directory or front panel. This realm is a name given to users so they know which username and password to send.

Scope

This directive applies to Global and per-directory ACFs.

The **AuthType**, **AuthUserFile**, and **AuthGroupFile** directives must accompany this directive for user authentication to work properly.

Syntax

```
AuthName name
```

`name` is a short name that describes this authorization realm. `name` can contain spaces.

Filename

```
access.cfg
```

Default

There is no default.

Example

The following directive sets the authorization name of this directory or front panel to `Research Group`:

```
AuthName Research Group
```

AuthType

The **AuthType** directive sets the type of authorization used in this directory or front panel.

Scope

This directive applies to Global and per-directory ACFs.

The **AuthName**, **AuthUserFile**, and **AuthGroupFile** directives must accompany this directive for user authentication to work properly.

Syntax

```
AuthType type
```

`type` is the authentication type to use for this directory or front panel. Currently, only the value `Basic` is supported for `type`.

Filename

`access.cfg`

Default

There is no default.

Example

The following directive sets the authorization type of this directory to `Basic`:

```
AuthType Basic
```

AuthUserFile

The **AuthUserFile** directive specifies the file to use as a list of users and passwords for user authentication.

Note ***You can manage your password files by selecting*** **Project»Internet Toolkit»Internet Toolkit Configuration…** ***and choosing*** **HTTP (Web Server)** ***from the drop-down menu in the*** **Internet Toolkit Configuration** ***dialog box. Click the*** **User File Editor** ***button to display the*** **Password User File Editor** ***dialog box.***

Scope

This directive applies to Global and per-directory ACFs.

The **AuthName**, **AuthType**, and **AuthGroupFile** directives must accompany this directive for user authentication to work properly.

Syntax

```
AuthUserFile path
```

`path` is the absolute path of a user file created with the HTTP Password Support VI.

Filename

`access.cfg`

Default

There is no default.

Example

The following directive sets the authorization user file for this directory or front panel to `/c/labview/internet/http/conf/passwd.txt`:

```
AuthUserFile /c/labview/internet/http/conf/passwd.txt
```

AuthGroupFile

The **AuthGroupFile** directive specifies the file to use as a list of user groups for user authentication.

☞ **Note** ***You can manage your group files by selecting*** **Project»Internet Toolkit»Internet Toolkit Configuration…** ***and choosing*** **HTTP (Web Server)** ***from the drop-down menu in the*** **Internet Toolkit Configuration** ***dialog box. Click the*** **Group File Editor** ***button to display the*** **Password Group File Editor** ***dialog box.***

Scope

This directive applies to Global and per-directory ACFs.

The **AuthName**, **AuthType**, and **AuthUserFile** directives must accompany this directive for user authentication to work properly.

Syntax

```
AuthGroupFile path
```

`path` is the absolute path of a group file to use in this directory or front panel.

Filename

`access.cfg`

Default

There is no default.

Example

The following directive sets the authorization group file for this directory or front panel to `/c/labview/internet/http/conf/group.txt`:

```
AuthGroupFile /c/labview/internet/http/conf/group.txt
```

Limit

The **Limit** sectioning directive controls which clients can access a directory or front panel.

Scope

This directive applies to Global and per-directory ACFs.

The **AuthName**, **AuthType**, and **AuthUserFile** directives must accompany this directive for user authentication to work properly.

Syntax

Opening directive:

```
<Limit meth1 meth2 ... methn>
```

Each `meth` is one of the following methods:

- `GET` (used in the **Directory** section)—Clients can retrieve documents and execute Common Gateway Interface (CGI) programs.
- `POST` (used in the **Directory** section)—Clients can use `POST` CGIs.
- `Snapshot` (used in the **Panel** section)—Clients can receive static images of VI front panels.
- `Monitor` (used in the **Panel** section)—Clients can receive server-push animations of VI front panels.

Closing directive:

```
</Limit>
```

You can use only the following directives inside the **Limit** section:

- **Order**
- **Deny**
- **Allow**
- **Require**
- **Satisfy**
- **Referer**
- **OnDeny**

Filename

`access.cfg`

Example

The following directive specifies that the only clients that can use the `.snap` command on this front panel must be from `.natinst.com` and authenticated group developers:

```
<Limit Snapshot>
order deny,allow
deny from all
allow from .natinst.com
require group developers
</Limit>
```

Order

The **Order** directive determines the order in which the **Deny** and **Allow** directives are evaluated within a **Limit** section.

Scope

You can use this directive only within **Limit** sections.

Syntax

```
order ord
```

`ord` is one of the following:

- `deny,allow`—In this case, you first evaluate all the **Deny** directives and then all the **Allow** directives. This order makes the **Deny** directives the default and the **Allow** directives the exceptions.
- `allow,deny`—In this case, you first evaluate all the **Allow** directives and then all the **Deny** directives. This order makes the **Allow** directives the default and the **Deny** directives the exceptions.
- `mutual-failure`—In this case, you specify hosts to allow or deny. Any host that appears on the **Allow** list is allowed, and any host on the **Deny** list is denied. If a host does not appear on either list, it is denied.

Filename

`access.cfg`

Default

If no order is given, the server assumes the following:

```
order deny,allow
```

Example

The following directive specifies the order `deny,allow` for this directory. In this command, the server first evaluates the **Deny** directive for the `GET` method, which denies all access to this directory. Then, the server evaluates the **Allow** directive, which allows clients from `.natinst.com` to `GET` files from this directory.

```
<Limit GET>
order deny,allow
deny from all
allow from .natinst.com
</Limit>
```

Deny

The **Deny** directive determines which hosts can access a given directory with a given method.

Scope

You can use this directive only within **Limit** sections.

Syntax

`deny from host1 host2 ... host`*n*

`host` is one of the following:

- A domain name—You can specify a domain name, such as `.natinst.com`. The host name must end in a domain name to deny.
- A host name—You can specify a full host name, such as `eagle.natinst.com`.
- A partial IP address—You can specify the first 1–3 octets of an IP address for subnet restriction.

☞ **Note** ***Because comparison is done by string matching, you must use trailing dots to match octets. For example,*** `130.164.1` ***matches host*** `130.164.12.18`***, but*** `130.164.1.` ***does not.***

- The keyword `all`—You can specify the keyword `all` to deny all hosts.

Filename

`access.cfg`

Default

The default is to restrict none.

Example

The following directive specifies the order `deny,allow` for this directory. In this command, the server first evaluates the **Deny** directive for the `GET` method, which denies all access to this directory. Then, the server evaluates the **Allow** directive, which allows clients from `.natinst.com` to `GET` files from this directory.

```
<Limit GET>
order deny,allow
deny from all
allow from .natinst.com
</Limit>
```

Allow

The **Allow** directive determines which hosts can access a given directory with a given method.

Scope

You can use this directive only within **Limit** sections.

Syntax

`allow from host1 host2 ... host`*n*

`host` is one of the following:

- A domain name—You can specify a domain name, such as `.natinst.com`. The host name must end in a domain name to deny.
- A host name—You can specify a full host name, such as `eagle.natinst.com`.
- A partial IP address—You can specify the first 1–3 octets of an IP address for subnet restriction.

☞ **Note** ***Because comparison is done by string matching, you must use trailing dots to match octets. For example,*** `130.164.1` ***matches host*** `130.164.12.18`***, but*** `130.164.1.` ***does not.***

- The keyword `all`—You can specify the keyword `all` to deny all hosts.

Filename

`access.cfg`

Default

The default is to allow from all.

Example

The following directive specifies the order `deny,allow` for this directory. In this command, the server first evaluates the **Deny** directive for the `GET` method, which denies all access to this directory. Then, the server evaluates the **Allow** directive, which allows clients from `.natinst.com` to `GET` files from this directory.

```
<Limit GET>
order deny,allow
deny from all
allow from .natinst.com
</Limit>
```

Require

The **Require** directive determines which authenticated users can access a given directory or front panel with a given method.

Scope

You can use this directive only within **Limit** sections.

Syntax

`require entity en1 en2 ... en`*n*

`en` are entity names, separated by spaces.

`entity` is one of the following:

- `user`—Restricts access to this directory or front panel, with the given methods, to only the named users.
- `group`—Restricts access to this directory or front panel, with the given methods, to only the named group.
- `valid-user`—All the users defined in the **AuthUserFile** can access once they provide a valid password.

Filename

`access.cfg`

Default

There is no default.

Example

The following directive specifies the order `deny,allow` and the **Require** directive for this directory. In this command, the server first evaluates the **Deny** directive for the `GET` method,

which denies all access to this directory. Then, the server evaluates the **Allow** directive, which allows clients from `.natinst.com` to `GET` files from this directory. Finally, the server evaluates the **Require** user authentication directive and allows only users named `bob` or users in the group `developers` access to this directory.

```
<Limit GET>
order deny,allow
deny from all
allow from .natinst.com
require user bob
require group developers
</Limit>
```

Referer

The **Referer** directive can be used to force users to enter a document from a specified path instead of jumping in at random. It allows the Webmaster to specify an exact match or wildcard expression to match the `Referer: HTTP` header. Refer to the **OnDeny** directive as a way to send the browser to the correct entry point.

☞ **Note** ***There is nothing to prevent a user from accessing the directory if they are able to modify the*** `Referer: header` ***that is sent with their browser.***

Scope

You can use this directive only within **Limit** sections.

Syntax

The **Referer** is a prefix to the standard **Allow** and **Deny** directives.

```
referer deny from URL
referer allow from URL
```

`URL` is a wildcard expression that specifies a URL.

Filename

`access.cfg`

Default

The default is to allow all **Referer** fields.

Example

The following example limits users to entering this directory through the `/test/` URLs. The `*` allows any port number or any directory beyond `/test/`. The **OnDeny** directive sends the browser to `http://my.server.com/test/`.

```
<Limit GET POST>
order deny,allow
deny from all
referer allow from http://my.server.com:*/test/*
OnDeny http://my.server.com/test/
</Limit>
```

Satisfy

The **Satisfy** directive determines the access to a directory if you use both the **Allow** and **Require** directives.

Scope

You can use this directive only within **Limit** sections.

Syntax

```
Satisfy all | any
```

`all` specifies that the user must satisfy both the **Allow** and **Require** directives to gain access to the directory.

`any` specifies that the user must satisfy only one of the specified **Allow** or **Require** directives.

Filename

`access.cfg`

Default

The default is `all`.

Example

The following directive restricts access for this directory to users from `.natinst.com` or users named `bob`. In this command, the server first evaluates the **Deny** directive, which denies all access to this directory. Next, the server evaluates the **Allow** directive, which allows clients from `.natinst.com` to access files in this directory. Finally, the server evaluates the **Satisfy** directive, which allows access to any users from `.natinst.com`. If you do not meet the

Satisfy directive, the server requires user authentication and allows access only to users named bob.

```
<Limit GET POST>
order deny,allow
deny from all
allow from .natinst.com
require user bob
satisfy any
</Limit>
```

OnDeny

The **OnDeny** directive provides non-HTTP–based access control to a directory. This directive sends a browser that fails the **Limit** directive to a specified URL.

Scope

You can use this directive only within **Limit** sections.

Syntax

```
OnDeny URL
```

`URL` is the URL you redirect the browser to.

Filename

`access.cfg`

Default

The default is to let the normal HTTP 401 response deal with the authentication failure.

Example

The following directive redirects any user who connects from outside `.natinst.com` to the URL `http://www.natinst.com/deny.htm`:

```
<Limit GET>
order deny,allow
deny from all
allow from .natinst.com
OnDeny http://www.natinst.com/deny.htm
</Limit>
```

Server Configuration Directives

The Server Configuration directives control the operation of the server. For more information on the Server Configuration specification, refer to the *Server Configuration* section in Chapter 8, *Using the G Web Server*.

Port

The **Port** directive sets what port the server uses to listen for clients.

Syntax

```
port num
```

`num` is a number from 0–65535. Most ports below 1024 are reserved by the system. For example, port 80 is reserved for HTTP.

To use a port below 1024 on UNIX and Windows NT, you might need to run the server as root or system administrator.

You can use only one **Port** directive in the configuration file.

Default

If you do not specify a **Port**, the server assumes the following:

```
Port 80
```

Note ***If you do not specify a* Port *other than the default, you must run the server as a root or system administrator on the UNIX or Windows NT platforms.***

Examples

- The following directive instructs the server to listen on port `8080`:

  ```
  Port 8080
  ```

- The following directive instructs the server to try to listen on port `84`:

  ```
  Port 84
  ```

UseDNS

The **UseDNS** directive specifies whether the server tries to get the Domain Name System (DNS) format for Internet Protocol (IP) addresses when logging data. Turn this option OFF on computers that are not connected to a domain name server. You can turn this option OFF on computers that have DNS access to improve performance, but this affects the access control based on DNS names.

Syntax

```
UseDNS setting
```

`setting` is ON or OFF.

You can use only one **UseDNS** directive in the configuration file.

Default

If you do not specify to **UseDNS**, the server assumes the following:

```
UseDNS off
```

Example

The following directive instructs the server to use domain names when describing connections:

```
UseDNS on
```

ServerAdmin

The **ServerAdmin** directive provides your e-mail address to the server. The server uses this address to let people report error conditions.

Syntax

```
ServerAdmin address
```

`address` is an e-mail address.

Default

If you do not specify a **ServerAdmin** address, the server prints `no address` for reporting errors.

Example

The following directive prints errors with the address `webmaster@foo.com` as the contact person:

```
ServerAdmin webmaster@foo.com
```

ServerRoot

The **ServerRoot** directive specifies the directory location of the server files.

On startup, the server expects to find the Server Configuration File as `conf/lvhttp.cfg` in the **ServerRoot** directory. Other Server Configuration directives might use this directory to provide relative paths for locations of files.

Syntax

```
ServerRoot dir
```

`dir` is an absolute path of a directory on your server computer.

You can use only one **ServerRoot** directive in the server configuration file.

Default

If you do not specify a **ServerRoot** directory, the server uses the directory in which the **ServerRoot** is located, as follows:

```
ServerRoot <default directory>/internet/http
```

Examples

- The following directive sets the **ServerRoot** to the directory `C:/toolkits/internet/http` on Windows:

  ```
  ServerRoot /c/toolkits/internet/http
  ```

- The following directive sets the **ServerRoot** to the directory `Macintosh HD:Web Stuff` on MacOS. You must encode the spaces in the path by preceding them with a `\`.

  ```
  ServerRoot /Macintosh\ HD/Web\ Stuff
  ```

ServerName

The **ServerName** directive sets the host name the server returns when creating redirection URLs. You use this directive on systems where the host name returned is a DNS alias, such as `www.natinst.com`.

Syntax

```
ServerName FQDN
```

`FQDN` is the full host name, including domain name, returned as the server address.

Default

If you do not specify a **ServerName**, the server attempts to retrieve it through the `IP To Name` function.

Example

The following directive sets the server host name as `www.natinst.com`:

```
ServerName www.natinst.com
```

☞ **Note** ***The default* ServerName *for this host name is*** `webserver.natinst.com`***, as the*** `IP To Name` ***function would return.***

StartServers

The **StartServers** directive determines how many connection handlers the server launches. More handlers require more memory but result in fewer pending requests.

Syntax

```
StartServers number
```

`number` is between 1 and 8, inclusive.

Default

If you do not specify a number, the server launches four handlers.

Example

The following directive launches eight connection handlers at startup:

```
StartServers 8
```

TimeOut

The **TimeOut** directive sets the amount of time the server waits for a client to send its query once the client connects to the server. **TimeOut** also can set the maximum amount of time the server spends waiting for a client to accept information.

Syntax

```
TimeOut time
```

`time` is the amount of time, in seconds, the server waits.

You can use only one **TimeOut** directive in the configuration file.

Default

If you do not specify a **TimeOut**, the server assumes the following:

```
TimeOut 600
```

This sets the timeout to 10 minutes.

Example

The following directive sets the timeout to 20 minutes, a useful time for large files on slow networks:

```
TimeOut 1200
```

AccessConfig

The **AccessConfig** directive specifies the location of the Global ACF.

Syntax

```
AccessConfig file
```

`file` is the name of the Global ACF, either a full pathname or a partial pathname relative to the **ServerRoot**.

You can use only one **AccessConfig** directive in the configuration file.

Default

If you do not specify an **AccessConfig** file, the server assumes the following:

```
AccessConfig conf/access.cfg
```

Examples

- The following directive instructs the server to look for the Global ACF in the file `conf/access-global` in the **ServerRoot** directory:

  ```
  AccessConfig conf/access-global
  ```

- The following directive instructs the server to look for the Global ACF in the file `/c/admin/access`:

  ```
  AccessConfig /c/admin/access
  ```

ResourceConfig

The **ResourceConfig** directive specifies the location of the Server Resource Map Configuration file.

Syntax

```
ResourceConfig file
```

`file` is the name of the Server Resource Map Configuration file, either a full pathname or a partial pathname relative to the **ServerRoot**.

You can use only one **ResourceConfig** directive in the configuration file.

Default

If you do not specify a **ResourceConfig** file, the server assumes the following:

```
ResourceConfig conf/srm.cfg
```

Examples

- The following directive instructs the server to look for the Server Resource Map Configuration file in the file `conf/resources` in the **ServerRoot** directory:

  ```
  ResourceConfig conf/resources
  ```

- The following directive instructs the server to look for the Server Resource Map Configuration file in the file `/c/admin/resources`:

  ```
  ResourceConfig /c/admin/resources
  ```

TypesConfig

The **TypesConfig** directive specifies the location of the typing configuration file. This file determines how the server maps filename extensions to Multipurpose Internet Mail Extensions (MIME) types for HTTP/1.0 clients. You should not need to edit this directive.

Syntax

```
TypesConfig file
```

`file` is the name of the Server Resource Map Configuration file, either a full pathname or a partial pathname relative to the **ServerRoot**.

You can use only one **TypesConfig** directive in the configuration file.

Default

If you do not specify a **TypesConfig** file, the server assumes the following:

```
TypesConfig conf/mime.typ
```

Examples

- The following directive instructs the server to look for the types configuration in the file `conf/mimetypes` in the **ServerRoot** directory:

  ```
  TypesConfig conf/mimetypes
  ```

- The following directive instructs the server to look for the types configuration in the file `/c/admin/mimetypes`:

  ```
  TypesConfig /c/admin/mimetypes
  ```

CGICacheTime

The **CGICacheTime** directive instructs the server how long a CGI Read Request VI waits before it times out.

Syntax

```
CGICacheTime time
```

`time` is the number of seconds the CGI Read Request VI waits for a request before it times out. A `time` of `-1` indicates to never time out.

You can use only one **CGICacheTime** directive in the configuration file.

Default

If you do not specify a **CGICacheTime**, the server assumes the following:

```
CGICacheTime 60
```

Examples

- To instruct the CGI Read Request VI to wait 1 minute for an incoming request before timing out, use the following command:

  ```
  CGICacheTime 60
  ```

- To instruct the CGI Read Request VI to never time out, use the following command. Any CGIs waiting with this timeout remain in memory until the server stops.

  ```
  CGICacheTime –1
  ```

ErrorLog

The **ErrorLog** directive determines the file to which the server logs errors and status messages it encounters. This directive logs the following information:

- Messages for Startup and Halt
- Clients that time out and/or abort
- Access files that attempt to override unauthorized directives
- Problems with the server
- Problems with User Authentication configuration
- Error messages that state a file does not exist

Syntax

```
ErrorLog file
```

`file` is the name of the Server Resource Map Configuration file, either a full pathname or a partial pathname relative to the **ServerRoot**.

You can use only one **ErrorLog** directive in the configuration file.

Default

If you do not specify an **ErrorLog**, the server assumes the following:

```
ErrorLog logs/error.log
```

Examples

- The following directive logs errors to the file `logs/errors` in the **ServerRoot** directory:

  ```
  ErrorLog logs/errors
  ```

- The following directive logs errors to the file `/tmp/errors`:

  ```
  ErrorLog /tmp/errors
  ```

- The following directive turns off error logging:

  ```
  ErrorLog /dev/null
  ```

☞ **Note** `/dev/null` ***is a special path known by the server. This path also works on non-UNIX platforms.***

TransferLog

The **TransferLog** directive determines where the server records client accesses.

Each line of the **TransferLog** log file format consists of the following code:

```
host - authuser [DD/Mon/YYYY:hh:mm:ss sdddd] "request" ddd bbbb
"opt_referer" "opt_agent"
```

- `host`—DNS name or IP number of the remote client
- `authuser`—Username, if sent for authentication, otherwise the symbol (`-`)
- `DD`—Day
- `Mon`—Month, calendar name
- `YYYY`—Year
- `hh`—Hour, 24-hour format according to the system time zone
- `mm`—Minutes
- `ss`—Seconds
- `sdddd`—Time offset from GMT, written as a sign (`+/-`) followed by four digits
- `request`—First line of the HTTP request the client sends
- `ddd`—Status code the server returns or (`-`) if not available
- `bbbb`—Total number of bytes sent, not including the HTTP/1.0 header, or (`-`) if not available
- `opt_referer`—Referer field if supplied and if **LogOptions** is `Combined`
- `opt_agent`—User agent field if supplied and if **LogOptions** is `Combined`

You can determine the name of the file accessed through `request`.

Syntax

```
TransferLog file
```

`file` is the name of the Server Resource Map Configuration file, either a full pathname or a partial pathname relative to the **ServerRoot**.

You can use only one **TransferLog** directive in the configuration file.

Default

If you do not specify a **TransferLog**, the server assumes the following:

```
TransferLog logs/access.log
```

Examples

- The following directive logs the transfers to the file `logs/transfers` in the **ServerRoot** directory:

  ```
  TransferLog logs/tranfers
  ```

- The following directive logs the transfers to the file `/tmp/transfers`:

  ```
  TransferLog /tmp/transfers
  ```

- The following directive turns off transfer logging:

  ```
  TransferLog /dev/null
  ```

☞ **Note** `/dev/null` ***is a special path known by the server. This path also works on non-UNIX platforms.***

AgentLog

The **AgentLog** directive determines where the server records client software. The **UserAgent** header, if present, is recorded in the format the client sends. If **LogOptions** is set to `Combined`, this directive does nothing.

All clients can use this directive for statistical purposes and to track protocol violations. The first whitespace-delimited word must be the software product name, with an optional slash and version designator. Other products that form part of the user agent can be written as separate words.

User AgentLog Example

The timestamp has the same format as in `access.log`.

```
[02/Oct/1996:14:48:07 -0500] Mozilla/2.0 (Macintosh; I; 68K)
[02/Oct/1996:15:01:29 -0500] Mozilla/3.0Gold (Win95; I)
```

```
[02/Oct/1996:18:23:51 -0500] Mozilla/2.0 (compatible; MSIE 3.0;
Windows 95)
[03/Oct/1996:15:47:17 -0500]  Lynx/2.3.7 BETA  libwww/2.14
```

Syntax

`AgentLog file`

`file` is the name of the file to which user agents log information, either a full pathname or a partial pathname relative to the **ServerRoot**.

You can use only one **AgentLog** directive in the configuration file.

Default

If you do not specify an **AgentLog**, the server assumes the following:

`AgentLog logs/agent.log`

Examples

- The following directive logs user agents to the file `logs/agents` in the **ServerRoot** directory:

  `AgentLog logs/agents`

- The following directive logs user agents to the file `/tmp/agents`:

  `AgentLog /tmp/agents`

- The following directive turns off user agent logging:

  `AgentLog /dev/null`

Note `/dev/null` ***is a special path known by the server. This path also works on non-UNIX platforms.***

RefererLog

The **RefererLog** directive determines where the server records refer. The **Referer** header, if present, is recorded in the following format:

`[Time Stamp] URI -> Document`

where `URI` is the Universal Resource Identifier for the document that references the server, and `Document` is the virtual path to the requested document. If **LogOptions** is set to `Combined`, this directive does nothing.

User RefererLog Example

The timestamp has the same format as in `access.log`.

```
[02/Oct/1996:18:22:08 -0500] http://cerberus/htdocs/ ->
/htdocs/images/logo.gif
```

```
[02/Oct/1996:18:22:11 -0500] http://cerberus/htdocs/ ->
/htdocs/config/config.htm
```

```
[02/Oct/1996:18:22:12 -0500]
http://cerberus/htdocs/config/config.htm ->
/htdocs/config/access/access.htm
```

Syntax

```
RefererLog file
```

`file` is the name of the file to which referers are logged, either a full pathname or a partial pathname relative to the **ServerRoot**.

You can use only one **RefererLog** directive in the configuration file.

Default

If you do not specify a **RefererLog**, the server assumes the following:

```
RefererLog logs/referer.log
```

Examples

- The following directive logs referers to the file `logs/referers` in the **ServerRoot** directory:

  ```
  RefererLog logs/referers
  ```

- The following directive logs referers to the file `/tmp/referers`:

  ```
  RefererLog /tmp/referers
  ```

- The following directive turns off referer logging:

  ```
  RefererLog /dev/null
  ```

Note `/dev/null` ***is a special path known by the server. This path also works on non-UNIX platforms.***

LogOptions

The **LogOptions** directive specifies your capability to switch between the Common Log File (CLF) format, where extra information goes to separate files, or a somewhat extended CLF format, where extra information is logged at the end of the normal CLF information. The **LogOptions** directive also determines your capability to change other aspects of server logging.

Because some log analyzers do not accept the **LogOptions** `Combined` format, you should check before you switch your files.

Combined Log Format Example

```
cerberus.natinst.com - - [03/Oct/1996:17:56:08 -0500] "GET / HTTP/1.0"
200 1564 - "Mozilla/2.0 (compatible; MSIE 3.0; Windows 95)"
```

```
cerberus.natinst.com - - [03/Oct/1996:17:56:21 -0500] "GET
/images/httpsrvr.gif HTTP/1.0" 200 1032 "http://cerberus/"
"Mozilla/2.0 (compatible; MSIE 3.0; Windows 95)"
```

```
cerberus.natinst.com - - [03/Oct/1996:17:56:25 -0500] "GET
/images/lvpower.gif HTTP/1.0" 200 1686 "http://cerberus/"
"Mozilla/2.0 (compatible; MSIE 3.0; Windows 95)"
```

```
cerberus.natinst.com - - [03/Oct/1996:17:56:25 -0500] "GET
/cgi-bin/counter.vi?index.html HTTP/1.0" 200 988 "http://cerberus/"
"Mozilla/2.0 (compatible; MSIE 3.0; Windows 95)"
```

☞ **Note** ***The preceding code includes extra* Refer *and* UserAgent *fields at the end of each command.***

Syntax

`LogOption opt1 opt2 ... opt`*n*

`opt` is one of the following:

- `Combined`—With this **LogOption**, the server does not open a **RefererLog** or **AgentLog**. Because these files are not open, the server logs the information normally logged to those files in the **TransferLog**.
- `Separate`—With this **LogOption**, the server uses separate log files for the **RefererLog** and **AgentLog**.
- `Date`—With this **LogOption**, the `Date` section of the **TransferLog** is added to the beginning of each line of the **RefererLog** and **AgentLog** to correlate the various files. This option is valid only with `Separate`.

Default

If you do not specify **LogOption**, the server assumes the following:

```
LogOptions Separate
```

Examples

- The following directive generates separate log files with date added:

  ```
  LogOptions Date
  ```

- The following directive generates a single log file that contains all the information:

  ```
  LogOptions Combined
  ```

PanelImageType

The **PanelImageType** directive specifies the default type of front panel images. Use this directive when no type is specified on a `.snap` or `.monitor` URL.

Syntax

```
PanelImageType type
```

`type` is the image type, either Joint Photographic Experts Group (JPEG) or Portable Network Graphics (PNG).

You can use only one **PanelImageType** directive in the configuration file.

Default

If you do not specify **PanelImageType**, the server assumes the following:

```
PanelImageType JPEG
```

Example

The following directive sets the default image type of front panels to PNG:

```
PanelImageType PNG
```

CheckPNGSupport

The **CheckPNGSupport** directive determines whether to change the default image type specified by the **PanelImageType** directive to the PNG image type for browsers with built-in PNG support. Generally, front panel images look better when encoded in PNG format rather than in JPEG format.

Syntax

```
CheckPNGSupport setting
```

`setting` is ON or OFF.

You can use only one **CheckPNGSupport** directive in the configuration file.

Default

If you do not specify to **CheckPNGSupport**, the server assumes the following:

```
CheckPNGSupport on
```

Example

The following directive instructs the server not to check for PNG support of the browser:

```
CheckPNGSupport off
```

PanelImageDepth

The **PanelImageDepth** directive specifies the default color depth of front panel images. Use this directive when no depth is specified on a `.snap` or `.monitor` URL.

Syntax

```
PanelImageDepth depth
```

`depth` is the color depth of the image. `depth` can be 1, 4, 8, or 24 bits.

You can use only one **PanelImageDepth** directive in the configuration file.

Default

If you do not specify **PanelImageDepth**, the server assumes the following:

```
PanelImageDepth 8
```

Example

The following directive sets the default color depth of front panels images to 24, or millions of colors:

```
PanelImageDepth 24
```

PanelImageQuality

The **PanelImageQuality** directive specifies the default quality of front panel images. This might apply only to certain image types, such as JPEG. Use this directive when no quality is specified on a `.snap` or `.monitor` URL.

Syntax

```
PanelImageQuality quality
```

`quality` is the quality of the image in percent; 0–100, with 100 representing the best.

You can use only one **PanelImageQuality** directive in the configuration file.

Default

If you do not specify **PanelImageQuality**, the server assumes the following:

```
PanelImageQuality 80
```

Example

The following directive sets the default quality of front panels images to 50 percent:

```
PanelImageQuality 50
```

PanelImageRefresh

The **PanelImageRefresh** directive specifies the length of time a VI image is kept in the server image cache before it is refreshed. A short **PanelImageRefresh** time returns more up-to-date images for `.snap` requests.

Syntax

```
PanelImageRefresh time
```

`time` is the number of seconds during which front panel images are cached.

You can use only one **PanelImageRefresh** directive in the configuration file.

Default

If you do not specify **PanelImageRefresh**, the server assumes the following:

```
PanelImageRefresh 10
```

Example

The following directive instructs the server to cache front panel images for 2 seconds:

```
PanelImageRefresh 2
```

PanelImageCacheCompactTime

The **PanelImageCacheCompactTime** directive specifies how often to compact the cache of front panel images.

Syntax

```
PanelImageCacheCompactTime time
```

`time` is number of seconds after which the cache is compacted.

You can use only one **PanelImageCacheCompactTime** directive in the configuration file.

Default

If you do not specify **PanelImageCacheCompactTime**, the server assumes the following:

```
PanelImageCacheCompactTime 300
```

Example

The following directive compacts the front panel image cache every 10 minutes:

```
PanelImageCacheCompactTime 600
```

ServerPushRefresh

The **ServerPushRefresh** directive specifies the default refresh rate for server-push animations. This value is used when no refresh is specified on a `.spool` or `.monitor` URL.

Syntax

```
ServerPushRefresh time
```

`time` is the number of seconds between each image.

You can use only one **ServerPushRefresh** directive in the configuration file.

Default

If you do not specify **ServerPushRefresh**, the server assumes the following:

```
ServerPushRefresh 10.0
```

Example

The following directive sets the default server-push refresh rate at once every 5 seconds:

```
ServerPushRefresh 5.0
```

ServerPushLifespan

The **ServerPushLifespan** directive specifies the lifespan of server-push animations. This value is used when no lifespan is specified on a `.spool` or `.monitor` URL.

Syntax

```
ServerPushLifespan time
```

`time` is the number of seconds before a server-push animation is terminated. A negative number indicates that only the client can stop the animation.

You can use only one **ServerPushLifespan** directive in the configuration file.

Default

If you do not specify **ServerPushLifespan**, the server assumes the following:

```
ServerPushLifespan 300
```

Examples

- The following directive sets the default server-push length of animation to 1 minute:

  ```
  ServerPushLifespan 60
  ```

- The following directive sets the default server-push length of animation to 0 seconds, which returns a static image:

  ```
  ServerPushLifespan 0
  ```

- The following directive sets the default server-push length of animation to infinity, which causes the server to wait for the client to stop the animation:

  ```
  ServerPushLifespan –1
  ```

ServerPushMaxConnections

The **ServerPushMaxConnections** directive specifies the maximum number of server-push animations the server returns at a time. When **ServerPushMaxConnections** server-push animations are running and a new request comes in, the animation older than **ServerPushMinTime** seconds stops and the new animation starts. If none of the animations are older than **ServerPushMinTime**, the server returns a static image for this request.

Syntax

```
ServerPushMaxConnections num
```

`num` is the maximum number of parallel server-push connections.

You can use only one **ServerPushMaxConnections** directive in the configuration file.

Default

If you do not specify **ServerPushMaxConnections**, the server assumes the following:

```
ServerPushMaxConnections 40
```

Example

The following directive sets the maximum number of parallel server-push animations to 10:

```
ServerPushMaxConnections 10
```

ServerPushMinTime

The **ServerPushMinTime** directive specifies the minimum number of seconds a server-push animation must exist before the server can stop it prematurely to service another server-push request. When **ServerPushMaxConnections** server-push animations are running and a new request comes in, the animation older than **ServerPushMinTime** seconds stops and the new animation starts. If none of the animations are older than **ServerPushMinTime**, the server returns a static image for this request.

Syntax

```
ServerPushMinTime time
```

`time` is the number of seconds before which the server can stop a server-push animation.

You can use only one **ServerPushMinTime** directive in the configuration file.

Default

If you do not specify **ServerPushMinTime**, the server assumes the following:

```
ServerPushMinTime 240
```

Example

The following directive sets the time before which a server-push animation can be prematurely stopped to 1 minute:

```
ServerPushMinTime 60
```

Server Resource Map Configuration Directives

The Server Resource Map Configuration directives control the presentation of the server documents. For more information on the Server Resource Map Configuration specification, refer to the *Server Resource Map Configuration* section in Chapter 8, *Using the G Web Server*.

DocumentRoot

The **DocumentRoot** directive determines the directory from which the server serves files.

If you want to serve files outside this directory, you can use the **Alias** directive.

Syntax

```
DocumentRoot dir
```

`dir` is the absolute path of the directory you want documents to be served from. The path is in UNIX format, which uses the path separator `/`.

You can use only one **DocumentRoot** directive in the configuration file.

Filename

```
srm.cfg
```

Default

If you do not specify a **DocumentRoot**, the server assumes the following:

```
DocumentRoot internet/http/htdocs
```

Examples

- The following directive sets the **DocumentRoot** to the directory `C:\LABVIEW\HOME` on Windows:

  ```
  DocumentRoot /c/labview/home
  ```

- The following directive sets the **DocumentRoot** to the directory `Macintosh HD:LabVIEW Folder:Home` on MacOS. You must encode the spaces in the path by preceding them with a `\`.

  ```
  DocumentRoot /Macintosh\ HD/LabVIEW\ Folder/Home
  ```

AccessFileName

The **AccesFileName** directive determines the name of the file the server checks to find directory ACFs. When returning documents to a client, the server looks for ACFs in the document directory and in the parent directories.

Syntax

```
AccessFileName file
```

`file` is a filename.

You can use only one **AccessFileName** directive in the configuration file.

Filename

`srm.cfg`

Default

If you do not specify an **AccessFileName**, the server assumes the following:

```
AccessFileName htaccess.txt
```

Example

The following directive sets the **AccessFileName** to `.htaccess`:

```
AccessFileName .htaccess
```

Redirect

The **Redirect** directive creates a virtual document on your server and redirects any access to the document to a new URL.

Syntax

```
Redirect virtual URL
```

`virtual` is the translated location that triggers a redirect.

`URL` is the URL of the new document.

You can use several **Redirect** directives in the configuration file.

Filename

`srm.cfg`

Default

There are no default **Redirect** directives.

Example

The following directive requests the document `/files` be redirected to the new location `ftp://ftp.natinst.com`:

```
Redirect /files ftp://ftp.natinst.com
```

Alias

The **Alias** directive creates a virtual document or directory on your server and uses a different file or directory name to satisfy access.

Syntax

```
Alias virtual path
```

`virtual` is the translated location of the file or directory.

`path` is the pathname of the file or directory used to fulfill the request. If `path` is a relative path, it is added to the end of **DocumentRoot**.

You can use several **Alias** directives in the configuration file.

Filename

`srm.cfg`

Default

By default, the server contains the following two **Alias** directives, which you can override:

```
Alias /htdocs/ <ServerRoot>/htdocs/
Alias /icons/ <ServerRoot>/icons/
```

Examples

- The following directive generates requests for `/images` from the directory `/ftp/pub/images`:

  ```
  Alias /images /ftp/pub/images
  ```

- The following directive generates requests for `/docs/` from the directory `<DocumentRoot>/documents/`:

  ```
  Alias /docs/ documents/
  ```

ScriptAlias

The **ScriptAlias** directive creates a virtual directory on your server. You satisfy access to the VIs in that virtual directory by returning the output of the CGI VI.

Syntax

```
ScriptAlias virtual path
```

`virtual` is the translated location of the CGI directory.

`path` is the pathname of the directory that contains the CGI VIs that fulfill the request. If `path` is a relative path, it is added to the end of **DocumentRoot**.

You can use several **ScriptAlias** directives in the configuration file.

Filename

`srm.cfg`

Default

There are no default **ScriptAlias** directives.

Examples

- The following directive fulfills the request `/cgi-bin/foo.vi` by running the VI `C:\labview\cgi-bin\foo.vi` on Windows:

  ```
  ScriptAlias /cgi-bin/ /c/labview/cgi-bin/
  ```

- The following directive fulfills the request `/cgi-bin/foo.vi` by running the VI `<DocumentRoot>/cgi-bin/foo.vi`:

  ```
  ScriptAlias /cgi-bin/ cgi-bin/
  ```

AddType

The **AddType** directive adds entries to the server default typing information and generates a particular extension type. This directive overrides any conflicting entries in the **TypesConfig** file.

Syntax

```
AddType type/subtype extension
```

`type/subtype` is the MIME-like type for the document.

`extension` is the filename extension to map to this type.

You can use several **AddType** directives in the configuration file.

Filename

`srm.cfg`

Default

The default types are located in the types configuration files.

Examples

- The following directive serves any file ending in `.doc` as type `text/plain`:

  ```
  AddType text/plain doc
  ```

- The following directive serves any file ending in `.txt` as type `text/plain`:

  ```
  AddType text/plain txt
  ```

AddEncoding

The **AddEncoding** directive specifies an encoding type for a document with a given extension.

To serve encoded documents to clients, the client must work with the given encoding method and the HTTP encoding extension.

Syntax

```
AddEncoding type extension
```

`type` is the encoding type for the document.

`extension` is the filename extension to map to this encoding.

You can use several **AddEncoding** directives in the configuration file.

Filename

`srm.cfg`

Default

There are no default encodings.

Example

The following directive marks any file ending in `.gz` as encoded using the `x-gzip` method:

```
AddEncoding x-gzip gz
```

Reference

Refer to the HTTP/1.0 Section 3.5 Content Codings in the Requests for Comments document HyperText Transfer Protocol (HTTP)—RFC 1945.

DefaultType

The **DefaultType** directive maps a file to a default type if the server cannot identify the type of a file through normal means.

Syntax

```
DefaultType type/subtype
```

`type/subtype` is the MIME-like type.

You can use only one **DefaultType** directive in the configuration file.

Filename

```
srm.cfg
```

Default

If no **DefaultType** is present, the server assumes the following:

```
DefaultType text/html
```

Example

The following directive returns files with an unknown extension type `application/octet-stream` (generic binary file):

```
DefaultType application/octet-stream
```

DirectoryIndex

The **DirectoryIndex** directive determines the file the server looks for to generate a pre-written index to a given directory.

Syntax

```
DiretoryIndex file
```

`file` is a filename.

You can use only one **DirectoryIndex** directive in the configuration file.

Filename

`srm.cfg`

Default

If no **DirectoryIndex** is present, the server assumes the following:

```
DirectoryIndex index.htm
```

Example

The following directive sets **DirectoryIndex** to `dir.htm`. This request for `/dir/` instructs the server to look for the file `DocumentRoot/dir/dir.htm`. If found, the server sends the file back to the client. Otherwise, the server creates and returns an index from the file system.

```
DirectoryIndex dir.htm
```

FancyIndexing

The **FancyIndexing** directive specifies whether you want fancy directory indexing that includes icons and file sizes or standard indexing for your file.

Turning fancy indexing off improves the performance of your server, especially on a slow file system.

Syntax

```
FancyIndexing setting
```

`setting` is ON or OFF.

You can use only one **FancyIndexing** directive in the configuration file.

Filename

`srm.cfg`

Default

If no **FancyIndexing** is present, the server assumes the following:

```
FancyIndexing off
```

Example

The following directive turns on fancy indexing:

```
FancyIndexing on
```

DefaultIcon

The **DefaultIcon** directive specifies what icon is shown in automatically generated directory listings for a file that contains no icon information.

Syntax

```
DefaultIcon location
```

`location` is the virtual path to the icon on your server.

You can use only one **DefaultIcon** directive in the configuration file.

Filename

`srm.cfg`

Default

If no **DefaultIcon** is present, the server does not display an icon.

Example

The following directive gives a file with no icon the icon `/icons/unknown.gif`:

```
DefaultIcon /icons/unknown.gif
```

ReadmeName

The **ReadmeName** directive specifies what filename the server looks for when indexing a directory. This directive adds a description paragraph to the end of the index it automatically generates.

Syntax

```
ReadmeName name
```

`name` is the name of the file the server looks for when trying to find a description file. The server first looks for `name.html` and `name.htm`. If found, the server displays the HTML inlined with its own index. If the server finds `name`, it includes the file as plain text.

You can use only one **ReadmeName** directive in the configuration file.

Filename

`srm.cfg`

Default

If no **ReadmeName** is present, the server does not look for a readme file.

Example

The following directive generates an index for the directory `/foo/` using the `name` files `/foo/README.html`, `/foo/README.htm`, and `/foo/README`:

```
ReadmeName README
```

HeaderName

The **HeaderName** directive determines what filename the server looks for when indexing a directory with a custom header.

Syntax

```
HeaderName name
```

`name` is the name of the file the server looks for when trying to find a description file. The server first looks for `name.html` and `name.htm`. If found, the server displays the HTML inlined with its own index. If the server finds `name`, it includes the file as plain text.

You can use only one **HeaderName** directive in the configuration file.

Filename

`srm.cfg`

Default

If no **HeaderName** is present, the server does not look for a header file.

Example

The following directive generates an index for the directory `/foo/`, using the `name` files `/foo/HEADER.html`, `/foo/HEADER.htm`, and `/foo/HEADER`:

```
HeaderName HEADER
```

AddIcon

The **AddIcon** directive determines the kind of icon the server uses to represent a given file type in a directory index.

Syntax

`AddIcon icon name1 name2 ... name`*n*

`icon` is a virtual path to an image file that is shown for files that match the pattern of names. Alternatively, this can be a group of the format `alt,icon`, where `alt` is the three-letter text tag given for an icon for non-graphical browsers, and `icon` is a virtual path.

`name` is `^^DIRECTORY^^` for directories, `^^BLANKICON^^` to specify the blank icon used to format the list properly, a file extension (`.html`), or a wildcard expression.

You can use several **AddIcon** directives in the configuration file.

Filename

`srm.cfg`

Default

There are no default icons.

Examples

- The following directive references the image `/icons/image.gif` for display next to the filename when indexing a file with the extensions `.gif`, `.jpg`, or `.xbm`:

  ```
  AddIcon /icons/image.gif .gif .jpg .xbm
  ```

- The following directive references `/icons/dir.gif` as the image for a subdirectory:

  ```
  AddIcon /icons/dir.gif ^^DIRECTORY^^
  ```

- The following directive references `/icons/sound.gif` as the image to show next to any `.au` sound file, with the textual `ALT` tag of `SND` for non-image clients:

  ```
  AddIcon (SND,/icons/sound.gif) *.au
  ```

AddIconByType

The **AddIconByType** directive determines the kind of icon to show for a given file type in a directory index.

Syntax

`AddIcon icon type1 type2 ... type`*n*

`icon` is a virtual path to an image file shown for files that match the pattern of names. Alternatively, this can be a group of the format `alt,icon`, where `alt` is the three-letter text tag given for an icon for non-graphical browsers, and `icon` is a virtual path.

`name` is a wildcard expression of the MIME-type to add to this icon.

You can use several **AddIconByType** directives in the configuration file.

Filename

`srm.cfg`

Default

There are no default icons.

Examples

- The following directive references `/icons/image.gif` as the image to show next to the filename when the server finds an image file:

  ```
  AddIconByType /icons/image.gif image/*
  ```

- The following directive references `/icons/sound.gif` as the image to show next to any sound file, with the textual `ALT` tag of `SND` for non-image clients:

  ```
  AddIconByType (SND,/icons/sound.gif) audio/*
  ```

AddIconByEncoding

The **AddIconByEncoding** directive determines the kind of icon to show for a given file type in a directory index.

Syntax

```
AddIconByEncoding icon name1 name2 ... namen
```

`icon` is a virtual path to an image file shown for files that match the pattern of names. Alternatively, this can be a group of the format `alt,icon`, where `alt` is the three-letter text tag given for an icon for non-graphical browsers, and `icon` is a virtual path.

`name` is a wildcard expression of the content-encoding to add to this icon.

You can use several **AddIconByEncoding** directives in the configuration file.

Filename

`srm.cfg`

Default

There are no default icons.

Examples

- The following directive references `/icons/compress.gif` as the image to show next to the filename if a file is compressed with the UNIX compress command:

  ```
  AddIconByEncoding /icons/compress.gif x-compress
  ```

- The following directive references `/icons/compress.gif` as the image to show next to UNIX compressed files, with the textual `ALT` tag of `CMP` for non-image clients:

  ```
  AddIconByEncoding (CMP,/icons/compress.gif) x-compress
  ```

IndexIgnore

The **IndexIgnore** directive specifies which files the server ignores when generating a directory index.

Syntax

```
IndexIgnore pat1 pat2 ... patn
```

`pat` is a wildcard expression against which files are matched.

You can use several **IndexIgnore** directives in the configuration file.

Filename

`srm.cfg`

Default

The current directory is the only entry default ignores.

Example

The following directive instructs the server to ignore files named `README`, `README.html`, and any file that starts with a period:

```
IndexIgnore */README */README.htm */.*
```

ErrorDocument

The **ErrorDocument** directive points the server to a file to send instead of the built-in error message.

Syntax

```
ErrorDocument type filename
```

`type` is one of the following:

- `302—REDIRECT`
- `400—BAD_REQUEST`
- `401—AUTH_REQUIRED`
- `403—FORBIDDEN`
- `404—NOT_FOUND`
- `500—SERVER_ERROR`
- `501—NOT_IMPLEMENTED`

`filename` is a path to a `text/html` file. If `filename` is relative, it is added to the end of **DocumentRoot**.

You can use only one **ErrorDocument** directive in the configuration file.

Filename

`srm.cfg`

Default

By default, the compiled error messages are used.

Examples

- `ErrorDocument 403 /error/fobidden.htm`
- `ErrorDocument 500 /error/srvrerr.htm`

B

Error Codes

This appendix lists the error codes the Internet Developers Toolkit for G VIs return, including the error number and a description. Each VI returns an error code that indicates whether the function was performed successfully. You also can find this table in the *Internet Toolkit VIs* online help.

Table B-1. Error Codes

Code	Type	Description
15110	FTP	110—Restart marker reply. In this case, the text is exact and not left to the particular implementation. It must read `MARK yyyy = mmmm`, where `yyyy` is the user-process data stream marker, and `mmmm` is the server equivalent marker. Notice the spaces between markers and `=`.
15120	FTP	120—Service ready in `nnn` minutes.
15125	FTP	125—Data connection already open; transfer starting.
15150	FTP	150—File status OK; about to open data connection.
15200	FTP	200—Command OK.
15202	FTP	202—Command not implemented; superfluous at this site.
15211	FTP	211—System status or system help reply.
15212	FTP	212—Directory status.
15213	FTP	213—File status.
15214	FTP	214—Help message. Provides information on how to use the server or the meaning of a particular non-standard command. This reply is useful only to the human user.
15215	FTP	215—`NAME` system type, where `NAME` is an official system name from the list in the Assigned Numbers document.
15220	FTP	220—Service ready for new user.
15221	FTP	221—Service closing control connection; logged out if appropriate.
15225	FTP	225—Data connection open; no transfer in progress.

Table B-1. Error Codes (Continued)

Code	Type	Description
15226	FTP	226—Closing data connection; requested file action successful.
15227	FTP	227—Entering passive mode (`h1,h2,h3,h4,p1,p2`).
15230	FTP	230—User logged in; proceed.
15250	FTP	250—Requested file action OK; completed.
15257	FTP	257—`PATHNAME` created.
15331	FTP	331—Username OK; need password.
15332	FTP	332—Need account for login.
15350	FTP	350—Requested file action pending further information.
15421	FTP	421—Service not available; closing control connection. This can be a reply to any command if the service knows it must shut down.
15425	FTP	425—Cannot open data connection.
15426	FTP	426—Connection closed; transfer aborted.
15450	FTP	450—Requested file action not taken; file unavailable.
15451	FTP	451—Requested action aborted; local error in processing.
15452	FTP	452—Requested action not taken; insufficient system storage.
15500	FTP	500—Syntax error; command unrecognized. This can include errors such as command line too long.
15501	FTP	501—Syntax error in parameters or arguments.
15502	FTP	502—Command not implemented.
15503	FTP	503—Bad sequence of commands.
15504	FTP	504—Command parameter not implemented.
15530	FTP	530—Not logged in.
15532	FTP	532—Need account for storing files.
15550	FTP	550—Requested action not taken; file unavailable.
15551	FTP	551—Requested action aborted; page type unknown.
15552	FTP	552—Requested file action aborted; exceeded storage allocation for current directory or dataset.

Table B-1. Error Codes (Continued)

Code	Type	Description
15553	FTP	553—Requested action not taken; filename not allowed.
16211	SMTP	211—System status or system help reply.
16214	SMTP	214—Help message. Provides information on how to use the receiver or the meaning of a particular non-standard command. This reply is useful only to the human user.
16220	SMTP	220—`<domain>` Service ready.
16221	SMTP	221—`<domain>` Service closing transmission channel.
16250	SMTP	250—Requested mail action OK; completed.
16251	SMTP	251—User not local; will forward to `<forward-path>`.
16354	SMTP	354—Start mail input; end with `<CRLF>.<CRLF>`.
16421	SMTP	421—`<domain>` Service not available; closing transmission channel. This can be a reply to any command if the service knows it must shut down.
16450	SMTP	450—Requested mail action not taken; mailbox unavailable.
16451	SMTP	451—Requested action aborted; local error in processing.
16452	SMTP	452—Requested action not taken; insufficient system storage.
16500	SMTP	500—Syntax error; command unrecognized. This can include errors such as command line too long.
16501	SMTP	501—Syntax error in parameters or arguments.
16502	SMTP	502—Command not implemented.
16503	SMTP	503—Bad sequence of commands.
16504	SMTP	504—Command parameter not implemented.
16550	SMTP	550—Requested action not taken; mailbox unavailable.
16551	SMTP	551—User not local; please try `<forward-path>`.
16552	SMTP	552—Requested mail action aborted; exceeded storage allocation.
16553	SMTP	553—Requested action not taken; mailbox name not allowed.
16554	SMTP	554—Transaction failed.

Table B-1. Error Codes (Continued)

Code	Type	Description
17200	HTTP	200—OK. The request has succeeded. The information returned with the response depends on the method used in the request.
17201	HTTP	201—Created. The request has been fulfilled and resulted in a new resource being created.
17202	HTTP	202—Accepted. The request has been accepted for processing, but the processing has not been completed.
17204	HTTP	204—No content. The server has fulfilled the request, but there is no new information to send back.
17300	HTTP	300—Multiple choices. The requested resource is available at one or more locations.
17301	HTTP	301—Moved permanently. The requested resource has been assigned a new permanent URL, and any future references to this resource should use that URL.
17302	HTTP	302—Moved temporarily. The requested resource resides temporarily under a different URL.
17304	HTTP	304—Not modified. The client has performed a conditional `GET` request and access is allowed, but the document has not been modified since the date and time specified in the `If-Modified-Since` field.
17400	HTTP	400—Bad request. The server could not understand the request because of incorrect syntax.
17401	HTTP	401—Unauthorized. The request requires user authentication.
17403	HTTP	403—Forbidden. The server understood the request but is refusing to fulfill it.
17404	HTTP	404—Not found. The server has not found anything matching the `Request-URL`.
17500	HTTP	500—Internal server error. The server encountered an unexpected condition that prevented it from fulfilling the request.
17501	HTTP	501—Not implemented. The server does not work with the functionality required to fulfill the request.
17502	HTTP	502—Bad gateway. The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request.

Table B-1. Error Codes (Continued)

Code	Type	Description
17503	HTTP	503—Service unavailable. The server is currently unable to deal with the request because of a temporary overload or maintenance of the server.
18000	URL	Invalid URL protocol.

C

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 manual 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 USA 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.

Country	Telephone	Fax
Australia	03 9879 5166	03 9879 6277
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Brazil	011 288 3336	011 288 8528
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Québec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 725 725 11	09 725 725 55
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 6120092	03 6120095
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
United Kingdom	01635 523545	01635 523154
United States	512 795 8248	512 794 5678

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: _____

_____

_____

_____

_____

Internet Developers Toolkit for G 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: *Internet Developers Toolkit for G Reference Manual*

Edition Date: June 1998

Part Number: 321392B-01

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

_____

_____

_____

_____

_____

_____

_____

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

_____

_____

_____

_____

_____

_____

_____

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

E-Mail Address _____

Phone (___) _____ Fax (___) _____

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

Fax to: Technical Publications
National Instruments Corporation
512 794 5678

Glossary

A

ACF	Access Configuration File.
active FTP mode	A File Transfer Protocol (FTP) mode in which the FTP server connects to the FTP client to make data connections. This default mode of operation might not work from inside strict Internet firewalls.
applet	An embedded Java application in an HTML document.
Application Programming Interface	A library of programming routines for performing a specific task.
ASCII	American Standard Code for Information Interchange.

B

block diagram	Pictorial description or representation of a program or algorithm.
browser	Software that displays World Wide Web pages on your computer.

C

client-side	Documents an application that resides on a browser system.
Common Gateway Interface (CGI)	Standard for external gateway programs to communicate with information servers, such as HTTP servers.
cookie	A token that identifies some other information.

D

directive	A line in a configuration file that consists of a keyword and some data and is used to configure the G Web Server.
Domain Name System (DNS)	A distributed name/address database used on the Internet.

E

environmental variables	Variables used by a CGI application that contain information about the server and the request.

F

File Transfer Protocol (FTP)	An application-level protocol used for accessing files or remote hosts on the Internet.
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.

G

G	The BridgeVIEW and LabVIEW graphical programming language.
Graphics Interchange Format (GIF)	A proprietary file format developed by CompuServe Information Service for the storage of 8-bit color, raster (bitmap) images. The format is based on the LZW data-compression algorithm.

H

hexadecimal	A base-16 number system.
HyperText Markup Language (HTML)	Language used to create hypertext documents. HTML tags can encode text attributes and specify links to documents or embedded images.
HyperText Transfer Protocol (HTTP)	A application-level protocol used to serve documents to remote clients on the Internet, or World Wide Web.

I

identity transliteration file	A mapping file that does not change the text it maps.
Image mode	The FTP transfer mode used for binary files.
Internet Protocol (IP)	An internetwork layer protocol that routes datagrams between hosts on the Internet.

J

Joint Photographic Experts Group (JPEG)	A standard designed for compressing color or gray-scale images of natural, real-world scenes.

K

Keyed Array	A string array data type in which elements are indexed by key strings.

L

LED	Light-emitting diode.

M

Multipurpose Internet Mail Extensions (MIME)	A specification for the interchange of multimedia e-mail among different computer systems that use Internet mail standards.

N

Network Virtual Terminal (NVT)	An imaginary device that provides a standard, networkwide, intermediate representation of a concanical terminal.

P

parse	To determine the syntactic structure of a sentence or string of symbols.
passive FTP mode	A File Transfer Protocol (FTP) mode in which the FTP client initiates data connections to the FTP server. Although this mode of operation works inside a strict Internet firewall, the default mode is *active FTP mode* because passive FTP mode does not work with some FTP servers.
Perl	A programming language that easily manipulates text, files, and processes.
Portable Network Graphics (PNG)	An extensible file format for the lossless, portable, well-compressed storage of raster (bitmap) images.
protocol	Set of rules that a computer uses to connect with other computers, specifying the format, timing, sequencing, and error checking for data transmission.

S

sectioning directive	Directive that can include other directives that are enclosed between opening and closing directive tags.
server-push animations	Animations that consist of a series of images an HTTP server sends over a period of time.
Simple Mail Transfer Protocol (SMTP)	Application-level protocol used for sending and forwarding e-mail on the Internet.
subVI	VI used in the block diagram of another VI; comparable to a subroutine.

T

tag	Specific HTML codes; specific word surrounded by less than (<) and greater than (>) signs.
target character set	A standard character set, such as ISO-Latin-1, used to encode an e-mail message or attachment for transmission over the Internet. A *virtual character set* is converted to a target character set by the process of *transliteration*.
Telnet	A TCP/IP protocol that consists of data and embedded control characters.

transliteration	Translating characters from one character set to another.
Transmission Control Protocol (TCP)	A transport layer protocol used on the Internet.

U

Universal Resource Locator (URL)	Form in which resources on the Internet are addressed on the World Wide Web.

V

virtual character set	A local character set used by a user to enter an e-mail message or attachment. A virtual character set is converted to a *target character set* by the process of *transliteration*.
virtual instrument (VI)	G program; so called because it models the appearance and function of a physical instrument.

W

wildcard expression	A string-matching expression in which a `?` represents an arbitrary character and `*` represents zero or more arbitrary characters.

Index

A

Access Configuration, 8-29 to 8-30
- directives, 8-29 to 8-30
- document directory control, 8-29
- Global Access Configuration file, 8-29
- Per-Directory Access Configuration file, 8-29

Access Configuration directives, A-1 to A-15
- Allow, A-11 to A-12
- AllowOverride, A-4
- AuthGroupFile, A-7
- AuthName, A-5
- AuthType, A-5 to A-6
- AuthUserFile, A-6 to A-7
- Deny, A-10 to A-11
- Directory, A-2 to A-3
- Limit, A-8 to A-9
- list of directives, 8-29 to 8-30
- OnDeny, A-15
- Order, A-9 to A-10
- Panel, A-3
- Referer, A-13 to A-14
- Require, A-12 to A-13
- Satisfy, A-14 to A-15

AccessConfig directive, A-20 to A-21
AccessFileName directive, A-37
Active CGI VIs, HTTP Server, 7-4
Active Connections, HTTP Server, 7-3
active FTP mode connection, 3-1
AddEncoding directive, A-40 to A-41
AddIcon directive, A-45
AddIconByEncoding directive, A-46 to A-47
AddIconByType directive, A-45 to A-46
AddType directive, A-39 to A-40
AgentLog directive, A-25 to A-26
Alias directive, A-38
Allow directive, A-11 to A-12
AllowOverride directive, A-4
animated front panel image
- .monitor URL, 8-6 to 8-8
  - examples, 8-8
  - purpose and use, 8-6
  - syntax, 8-6 to 8-7
- .spool URL, 8-8 to 8-10
  - examples, 8-9 to 8-10
  - purpose and use, 8-8
  - syntax, 8-8 to 8-9

anonymous FTP, 2-2, 3-1
ASCII character set, 4-3
ASCII mode, FTP, 3-1 to 3-2
AuthGroupFile directive, A-7
AuthName directive, A-5
AuthType directive, A-5 to A-6
AuthUserFile directive, A-6 to A-7

B

Build URL VI, 6-2
bulletin board support, C-1

C

Cached CGI VIs, HTTP Server, 7-4
case sensitivity, in configuration files, 8-28
CGI
- basic principles, 7-7 to 7-8
- definition, 8-10

CGI applications, 8-10 to 8-13
- application process (figure), 8-10
- building in G, 1-2
- CGI VI example, 8-13
- definition, 8-10
- environment variables (table), 8-12

GET method of supplying parameters, 8-11
POST method of supplying parameters, 8-11
structure of CGI application (figure), 8-13
CGI Parameter VIs, 8-14
CGI Utility VIs, 8-13 to 8-23
active, 7-4
cached, 7-4
categories, 8-13 to 8-14
CGI Parameter VIs, 8-14
File Path VIs, 8-15
HTML VIs, 8-17 to 8-23
HTML Document VI (example), 8-20
HTML Generic Tag VI (example), 8-20
HTML table block diagram, 8-18
HTML+Monitor VI (example), 8-19
HTML+String Array to Table VI (example), 8-19
list of VIs, 8-21 to 8-23
Keyed Array VIs, 8-16 to 8-17
Network VIs, 8-15 to 8-16
Panel Image VIs, 8-16
CGICacheTime directive, A-22 to A-23
character, definition, 4-2
character codes, 4-3
character sets, 4-2 to 4-4
definition, 4-3
ISO Latin-1, 4-3
Macintosh, 4-3 to 4-4
overview, 4-2 to 4-3
transliteration, 4-4
US-ASCII, 4-3
CheckPNGSupport directive, A-30
client state information, maintaining, 8-24 to 8-27
client-side cookies, 8-24 to 8-25
server-side cookies, 8-25 to 8-27
client-side cookies, 8-24 to 8-25
cookie VIs, 8-25
example, 8-25
using cookies, 8-24
comment lines, in configuration files, 8-28
Common Gateway Interface (CGI). *See* CGI; CGI applications; CGI Utility VIs.
configuration, 2-1 to 2-3
[ftp] configuration, 2-2
G Web Server
Access Configuration, 8-29 to 8-30
advanced options, 8-28 to 8-32
Configuration Directives, 8-30 to 8-31
Internet Toolkit Configuration dialog box, 7-4 to 7-5
rules for configuration files, 8-28
Server Configuration, 8-30 to 8-31
Server Resource Map Configuration, 8-31
[http] configuration, 2-3
Internet Toolkit Configuration dialog box, 2-1, 7-4 to 7-5
mass compiling directories, 2-1
[smtp] configuration, 2-2 to 2-3
Toolkit configuration, 2-1
configuration directives, A-1 to A-48
Access Configuration, A-1 to A-15
Allow, A-11 to A-12
AllowOverride, A-4
AuthGroupFile, A-7
AuthName, A-5
AuthType, A-5 to A-6
AuthUserFile, A-6 to A-7
Deny, A-10 to A-11
Directory, A-2 to A-3
Limit, A-8 to A-9
list of directives, 8-29 to 8-30
OnDeny, A-15
Order, A-9 to A-10
Panel, A-3

Referer, A-13 to A-14
Require, A-12 to A-13
Satisfy, A-14 to A-15
in configuration files, 8-28
Server Configuration, A-16 to A-35
AccessConfig, A-20 to A-21
AgentLog, A-25 to A-26
CGICacheTime, A-22 to A-23
CheckPNGSupport, A-30
ErrorLog, A-23 to A-24
list of directives, 8-30 to 8-31
LogOptions, A-28 to A-29
PanelImageCacheCompactTime, A-32
PanelImageDepth, A-30 to A-31
PanelImageQuality, A-31
PanelImageRefresh, A-31 to A-32
PanelImageType, A-29
Port, A-16
RefererLog, A-26 to A-27
ResourceConfig, A-21
ServerAdmin, A-17
ServerName, A-19
ServerPushLifespan, A-33
ServerPushMaxConnections, A-34
ServerPushMinTime, A-34 to A-35
ServerPushRefresh, A-32 to A-33
ServerRoot, A-18
StartServers, A-19
TimeOut, A-20
TransferLog, A-24 to A-25
TypesConfig, A-22
UserDNS, A-17
Server Resource Map Configuration, A-36 to A-48
AccessFileName, A-37
AddEncoding, A-40 to A-41
AddIcon, A-45
AddIconByEncoding, A-46 to A-47
AddIconByType, A-45 to A-46
AddType, A-39 to A-40
Alias, A-38
DefaultIcon, A-43
DefaultType, A-41
DirectoryIndex, A-41 to A-42
DocumentRoot, A-36
ErrorDocument, A-48
FancyIndexing, A-42 to A-43
HeaderName, A-44
IndexIgnore, A-47
list of directives, 8-31 to 8-32
ReadmeName, A-43 to A-44
Redirect, A-37 to A-38
ScriptAlias, A-39
wildcard expressions, A-1
cookie VIs
client-side cookie VIs, 8-25
server-side cookie VIs, 8-26 to 8-27
cookies
client-side cookies, 8-24 to 8-25
definition, 8-24
server-side cookies, 8-25 to 8-27
customer communication, xiv, C-1 to C-2

D

data transfer. *See* FTP; FTP VIs.
DefaultIcon directive, A-43
DefaultType directive, A-41
Deny directive, A-10 to A-11
detailed mode, HTTP Server, 7-3 to 7-4
directives. *See* configuration directives.
Directory directive, A-2 to A-3
DirectoryIndex directive, A-41 to A-42
documentation
conventions used in manual, xii
organization of manual, xi-xii
related documentation, xiii
DocumentRoot directive, A-36

E

electronic support services, C-1 to C-2
e-mail capabilities, 1-2
e-mail technical support, C-2
E-mail VIs, 4-1 to 4-6
    character sets, 4-2 to 4-4
        definition, 4-3
        ISO Latin-1, 4-3
        Macintosh, 4-3 to 4-4
        overview, 4-2 to 4-3
        transliteration, 4-4
        US-ASCII, 4-3
    getting started, 4-5 to 4-6
    high-level VIs, 4-2, 4-5
    intermediate VIs, 4-2
    low-level VIs, 4-1
    purpose and use, 4-1
environment variables for CGI applications (table), 8-12
error codes, B-1 to B-5
ErrorDocument directive, A-48
ErrorLog directive, A-23 to A-24

F

FancyIndexing directive, A-42 to A-43
fax and telephone support numbers, C-2
Fax-on-Demand support, C-2
File path format, in configuration files, 8-28
File Path VIs, 8-15
File Transfer Protocol. *See* FTP.
front panel images
    animated front panel image
        .monitor URL, 8-6 to 8-8
        .spool URL, 8-8 to 8-10
    formats, 8-4
    static front panel image (.snap URL), 8-4 to 8-6
    viewing VI front panels remotely, 7-6 to 7-7
FTP
    active connection, 3-1
    anonymous, 2-2, 3-1
    ASCII mode, 3-1 to 3-2
    Image mode, 3-1 to 3-2
    passive connection, 3-1
    purpose and use, 3-1
FTP configuration, 2-2
FTP technical support, C-1
FTP VIs, 3-1 to 3-5
    getting started, 3-4 to 3-5
    high-level, 3-2 to 3-3
        list of VIs, 3-3 to 3-4
    intermediate, 3-2
    low-level, 3-2

G

G Web Server, 7-1 to 7-9, 8-1 to 8-32
    CGI applications, 8-10 to 8-13
        application process (figure), 8-10
        CGI basics, 7-8 to 7-9
        CGI VI example, 8-13
        definition, 8-10
        environment variables (table), 8-12
        GET method of supplying parameters, 8-11
        POST method of supplying parameters, 8-11
        structure of CGI application (figure), 8-13
    CGI Utility VIs, 8-13 to 8-23
        categories, 8-13 to 8-14
        CGI Parameter VIs, 8-14
        File Path VIs, 8-15
        HTML VIs, 8-17 to 8-23
            HTML Document VI (example), 8-20
            HTML Generic Tag VI (example), 8-20

HTML table block diagram, 8-18
HTML+Monitor VI (example), 8-19
HTML+String Array to Table VI (example), 8-19
list of VIs, 8-21 to 8-23
Keyed Array VIs, 8-16 to 8-17
Network VIs, 8-15 to 8-16
Panel Image VIs, 8-16
configuration
Access Configuration, 8-29 to 8-30
advanced options, 8-28 to 8-32
Configuration Directives, 8-30 to 8-31
Internet Toolkit Configuration dialog box, 7-4 to 7-5
rules for configuration files, 8-28
Server Configuration, 8-30 to 8-31
Server Resource Map Configuration, 8-31
definition, 7-1
features, 7-1 to 7-2
getting started, 7-6 to 7-9
HTML, 7-5 to 7-6
HTTP Server, 7-2 to 7-4
maintaining client state information, 8-24 to 8-27
client-side cookies, 8-24 to 8-25
server-side cookies, 8-25 to 8-27
online examples, 7-8
purpose and use, 7-1 to 7-4
starting, 7-1
URLs for front panel images, 8-4 to 8-10
animated front panel image (.monitor URL), 8-6 to 8-8
animated front panel image (.spool URL), 8-8 to 8-10
front panel image formats, 8-4
static front panel image (.snap URL), 8-4 to 8-6
viewing VI front panels remotely, 7-6 to 7-7
working with VIs, 8-1 to 8-3
embedding images, 8-3
executing VIs, 8-2 to 8-3
viewing running VIs, 8-1 to 8-2
GET method of supplying CGI parameters, 8-11
Global Access Configuration file, 8-29

H

HeaderName directive, A-44
high-level VIs
E-mail VIs, 4-2, 4-5
FTP VIs, 3-2 to 3-3
HTML, 7-5 to 7-6
definition, 7-5
standards, 7-6
tags, 7-5 to 7-6
HTML VIs, 8-17 to 8-23
HTML Document VI (example), 8-20
HTML Generic Tag VI (example), 8-20
HTML table block diagram, 8-18
HTML+Monitor VI (example), 8-19
HTML+String Array to Table VI (example), 8-19
list of VIs, 8-21 to 8-23
HTTP (Hyper Text Transfer Protocol) configuration, 2-3
HTTP Server, 7-2 to 7-4
detailed mode
Active CGIs, 7-4
Active Connections, 7-3
Cached CGIs, 7-4
example (figure), 7-3
Log information, 7-4
Pending Connections, 7-3
Server Activity information, 7-3
Server-Push Connections, 7-3

simple mode, 7-2
STOP button, 7-2
HTTP Server Control VI, 7-1
Hyper Text Transfer Protocol (HTTP) configuration, 2-3
HyperText Markup Language (HTML). *See* HTML.

I

Image mode, FTP, 3-1 to 3-2
images
embedding, 8-3
panel. *See* front panel images.
IndexIgnore directive, A-47
intermediate VIs
E-mail VIs, 4-2
FTP VIs, 3-2
Internet Developers Toolkit for G
components, 1-1
configuration, 2-1 to 2-3
[ftp] configuration, 2-2
[http] configuration, 2-3
Internet Toolkit Configuration dialog box, 2-1
mass compiling directories, 2-1
[smtp] configuration, 2-1 to 2-3
purpose and use, 1-1 to 1-2
building CGI programs in G, 1-2
converting VIs to Internet applications, 1-2
sending e-mail and data files, 1-2
Internet Toolkit Configuration dialog box, 7-4 to 7-5
DocumentRoot field, 7-5
overview, 2-1
Port field, 7-5
Server Admin field, 7-4 to 7-5
UseDNS checkbox, 7-5
internet.ini file, 2-2
ISO Latin-1 character set, 4-3
ISO-8859-1 character set, 4-3

J

JPEG (Joint Photographic Experts Group) format, 8-4

K

Keyed Array VIs, 8-16 to 8-17

L

Limit directive, A-8 to A-9
Log information, HTTP Server, 7-4
LogOptions directive, A-28 to A-29
low-level VIs
E-mail VIs, 4-1
FTP VIs, 3-2

M

Macintosh character set, 4-3 to 4-4
manual. *See* documentation.
mass compiling directories, 2-1
.monitor URL, 8-6 to 8-8
examples, 8-8
purpose and use, 8-6
syntax, 8-6 to 8-7

N

negotiated options, 5-1 to 5-2
Network Virtual Terminal (NVT), 5-1
Network VIs, 8-15 to 8-16
NVT (Network Virtual Terminal), 5-1

O

OnDeny directive, A-15
Order directive, A-9 to A-10

P

Panel directive, A-3
Panel Image VIs, 8-16
PanelImageCacheCompactTime directive, A-32
PanelImageDepth directive, A-30 to A-31
PanelImageQuality directive, A-31
PanelImageRefresh directive, A-31 to A-32
PanelImageType directive, A-29
Parse URL VI, 6-2
passive connection for FTP, 3-1
PathDirective, in configuration files, 8-28
Pending Connections, HTTP Server, 7-3
Per-Directory Access Configuration file, 8-29
PNG image format, 8-4
Port directive, A-16
Portable Network Graphics (PNG) image format, 8-4
POST method of supplying CGI parameters, 8-11

R

ReadmeName directive, A-43 to A-44
Redirect directive, A-37 to A-38
Referer directive, A-13 to A-14
RefererLog directive, A-26 to A-27
Require directive, A-12 to A-13
ResourceConfig directive, A-21

S

Satisfy directive, A-14 to A-15
ScriptAlias directive, A-39
Server. *See* G Web Server; HTTP Server.
Server Activity, HTTP Server, 7-3
Server Configuration
    list of directives, 8-30 to 8-31
    purpose, 8-30
Server Configuration directives, A-16 to A-35
    AccessConfig, A-20 to A-21
    AgentLog, A-25 to A-26
    CGICacheTime, A-22 to A-23
    CheckPNGSupport, A-30
    ErrorLog, A-23 to A-24
    list of directives, 8-30 to 8-31
    LogOptions, A-28 to A-29
    PanelImageCacheCompactTime, A-32
    PanelImageDepth, A-30 to A-31
    PanelImageQuality, A-31
    PanelImageRefresh, A-31 to A-32
    PanelImageType, A-29
    Port, A-16
    RefererLog, A-26 to A-27
    ResourceConfig, A-21
    ServerAdmin, A-17
    ServerName, A-19
    ServerPushLifespan, A-33
    ServerPushMaxConnections, A-34
    ServerPushMinTime, A-34 to A-35
    ServerPushRefresh, A-32 to A-33
    ServerRoot, A-18
    StartServers, A-19
    TimeOut, A-20
    TransferLog, A-24 to A-25
    TypesConfig, A-22
    UserDNS, A-17
Server Resource Map Configuration
    list of directives, 8-31 to 8-32
    purpose, 8-31
Server Resource Map Configuration directives, A-36 to A-48
    AccessFileName, A-37
    AddEncoding, A-40 to A-41
    AddIcon, A-45
    AddIconByEncoding, A-46 to A-47
    AddIconByType, A-45 to A-46

AddType, A-39 to A-40
Alias, A-38
DefaultIcon, A-43
DefaultType, A-41
DirectoryIndex, A-41 to A-42
DocumentRoot, A-36
ErrorDocument, A-48
FancyIndexing, A-42 to A-43
HeaderName, A-44
IndexIgnore, A-47
list of directives, 8-31 to 8-32
ReadmeName, A-43 to A-44
Redirect, A-37 to A-38
ScriptAlias, A-39
ServerAdmin directive, A-17
ServerName directive, A-19
Server-Push Connections, HTTP Server, 7-3
ServerPushLifespan directive, A-33
ServerPushMaxConnections directive, A-34
ServerPushMinTime directive, A-34 to A-35
ServerPushRefresh directive, A-32 to A-33
ServerRoot directive, A-18
server-side cookies, 8-25 to 8-27
cookie VIs, 8-26 to 8-27
example, 8-26
using, 8-26
Simple Mail Transfer Protocol (SMTP) configuration, 2-2 to 2-3
simple mode, HTTP Server, 7-2
SMTP (Simple Mail Transfer Protocol) configuration, 2-2 to 2-3
SMTP Send Data VI, 4-5
SMTP Send File VI, 4-5
SMTP Send Message VI, 4-5
SMTP Send Message (Small) VI, 4-5
SMTP Send Multiple Attachments VI, 4-5
.snap URL, 8-4 to 8-6
examples, 8-6
syntax, 8-5
spaces, in configuration files, 8-28
.spool URL, 8-8 to 8-10
examples, 8-9 to 8-10
purpose and use, 8-8
syntax, 8-8 to 8-9
starting the G Web Server, 7-1
StartServers directive, A-19
static front panel image (.snap URL), 8-4 to 8-6
examples, 8-6
syntax, 8-5

T

technical support, C-1 to C-2
telephone and fax support numbers, C-2
Telnet connection
negotiated options, 5-1 to 5-2
Network Virtual Terminal, 5-1
Telnet Protocol, 5-1
Telnet VIs, 5-2 to 5-3
getting started, 5-3
list of VIs, 5-2 to 5-3
purpose and use, 5-2
TimeOut directive, A-20
TransferLog directive, A-24 to A-25
transliteration of character sets, 4-4
TypesConfig directive, A-22

U

Universal Resource Locators (URLs). *See* URL VIs; URLs.
URL VIs, 6-2
URLs
CGI Parameter VIs, 8-14
definition, 6-1

examples, 6-1
for front panel images, 8-4 to 8-10
animated front panel image (.monitor URL), 8-6 to 8-8
animated front panel image (.spool URL), 8-8 to 8-10
front panel image formats, 8-4
static front panel image (.snap URL), 8-4 to 8-6
US-ASCII character set, 4-3
UserDNS directive, A-17

V

VIs
CGI Utility VIs, 8-13 to 8-23
active, 7-4
cached, 7-4
categories, 8-13 to 8-14
CGI Parameter VIs, 8-14
File Path VIs, 8-15
HTML VIs, 8-17 to 8-23
Keyed Array VIs, 8-16 to 8-17
Network VIs, 8-15 to 8-16
Panel Image VIs, 8-16
converting to Internet applications, 1-2
cookie VIs
client-side cookie VIs, 8-25
server-side cookie VIs, 8-26 to 8-27
E-mail VIs, 4-1 to 4-6
high-level VIs, 4-2, 4-5
intermediate VIs, 4-2
low-level VIs, 4-1
purpose and use, 4-1
embedding images, 8-3
executing VIs, 8-2 to 8-3
File Path VIs, 8-15
FTP VIs, 3-1 to 3-5
high-level, 3-2 to 3-3
intermediate, 3-2
low-level, 3-2
HTML VIs, 8-17 to 8-23
HTML Document VI (example), 8-20
HTML Generic Tag VI (example), 8-20
HTML table block diagram, 8-18
HTML+Monitor VI (example), 8-19
HTML+String Array to Table VI (example), 8-19
list of VIs, 8-21 to 8-23
HTTP Server Control VI, 7-1
Keyed Array VIs, 8-16 to 8-17
Network VIs, 8-15 to 8-16
Panel Image VIs, 8-16
Parse URL VI, 6-2
Telnet VIs, 5-2 to 5-3
list of VIs, 5-2 to 5-3
purpose and use, 5-2
URL VIs, 6-2
viewing running VIs, 8-1 to 8-2

W

Web Server. *See* G Web Server.
white space, in configuration files, 8-28
wildcard expressions, A-1