z/OS

HTTP Server Planning, Installing, and Using

Version 5.3

SC34-4826-03

z/OS

HTTP Server Planning, Installing, and Using

Version 5.3

SC34-4826-03

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 681.

Fourth Edition (September 2002)

This edition applies to Version 5.3 of the IBM HTTP Server for z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. To tell us what you think, use the email address on our Web site Library page at URL:

`http://www.ibm.com/software/webservers/httpservers/doc53.html`

©Copyright International Business Machines Corporation 1998, 2001, 2002. All rights reserved.

U.S. Government Users Restricted Rights — Use, duplication, or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Summary of Changes for IBM HTTP Server Version 5.3 for z/OS xiii
SC34–4826–03 for z/OS Version 1 Release 4 . . . xiii
Security xiii
Messages xiii

Welcome! xv
Product information xv
Support services. xv

Part 1. Planning 1

Chapter 1. Planning for installation . . . 3
Requirements 3
APARs and service updates 3
Compatible and tested browser levels 3
IBM Software Development Kit 5
z/OS UNIX System Services 5
WebSphere Application Server 7
Considerations. 7
ABEND recovery performed by the Web server. . 7
Access control with z/OS RACF or other security products. 8
Fast Response Cache Accelerator 8
MVSDS DLL z/OS data set considerations . . . 9
Performance 9
Reporting programs used by the Web server . . 10
Security 10
Workload Management 11

Chapter 2. Migration considerations . . 13
Installation 13
Backing up existing configuration files 13
Using Version 5.3 configuration files 13
Removing IMW.SIMWMOD1 from your LNKLST and start procedure. 13
Reviewing Version 5.3 installation and customization information 13
IMWHTTPD program 14
Java™ support 15
Proxy server and caching proxy server 15
Security 15
Certificate and key management changes . . . 15
Certificate authority utility changes 16

Part 2. Installing 17

Chapter 3. Installing your secure server 19
Before you begin 19
Complete z/OS installation 19
Review Web server planning and migration information 19
Check for information updates on the Web . . . 19
Completing and customizing your Web server installation 20
Step 1. Set up IDs used by the Web server . . . 20
Step 2. Turn on program control for DLLs . . . 24
Step 3. Enable the Web server to use optional functions 25
Step 4. Make TCP/IP configuration adjustments as needed 28
Step 5. Copy and customize the Web server PROC 29
Step 6. Configure installed files by running setup.sh 30
Step 7. Configure installed files on your target system by running setuptgt.sh (optional) . . . 33
Step 8. Customize your Web server configuration file httpd.conf 35
Step 9. Customize your Web server environment variables file 35
Step 10. Set up security 36
Step 11. Set up proxy server support (optional) 36
Step 12. Install and configure WebSphere Application Server (optional) 36
What's next? 36

Part 3. Using 37

Chapter 4. Starting the server 39
Performance considerations when starting the Web server 39
Starting the server from the start-up PROC 39
Starting the server using the httpd command . . . 39
Starting the server from the z/OS UNIX shell . . . 39
Starting multiple instances of the server 40
Starting the server with Workload Management running. 40
Letting WLM start the queue server address space 40
Prestarting the queue server address space . . . 40
Restarting the server 41

Chapter 5. Viewing the Web server's default Front Page. 43

Chapter 6. Stopping the server 45

Part 4. Basic Configuration. 47

Chapter 7. Getting started 49
How do I start serving pages? 49
How do I serve directory listings? 49
How do I configure the Web server? 50
Editing the configuration file 50

Using the Configuration and Administration forms . . . 50
How do I control access to server administration functions? . . . 51
Controlling access to the Configuration and Administration forms . . . 51
Copying files . . . 51
What files should I back up? . . . 52

Chapter 8. Setting up a secure server 53

Security concepts . . . 55
What is a secure connection? . . . 55
What is encryption? . . . 56
What is authentication? . . . 57
What is SSL? . . . 58
What is a Public Key Infrastructure? . . . 58
What is LDAP? . . . 59
Security options for the HTTP Server . . . 61
Options for setting up secure connections . . . 61
SSL support for multiple IP addresses . . . 62
Support for specifying the encryption level to be used . . . 62
Certificate authorities supported by the HTTP Server . . . 62
Buying a certificate from an external commercial CA . . . 62
Acting as your own CA . . . 63
Encryption support for the HTTP Server. . . . 64
Supported public-private key sizes . . . 64
Supported SSL cipher specifications . . . 64
Hardware encryption . . . 65
Checklist for setting up a secure server . . . 68
Examples: Setting up secure connections. . . . 69
Important notes . . . 69
Which example should I use? . . . 70
Terms used in these examples . . . 73
Example 1A for gskkyman: Setting up secure connections using a self-signed server certificate (z/OS Version 1 Release 4 or later) . . . 75
Example 2A for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 4 or later) . . . 78
Example 3A for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 4 or later) . 84
Example 4A for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 4 or later) . . . 90
Example 1 for gskkyman: Setting up secure connections using a self-signed server certificate (z/OS Version 1 Release 3 or earlier) . . . 98
Example 2 for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 3 or earlier) . . . 101
Example 3 for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 3 or earlier) . . . 107
Example 4 for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 3 or earlier) . . . 113
Example 5 for RACDCERT: Setting up secure connections using a self-signed server certificate. 120
Example 6 for RACDCERT: Setting up secure connections using certificates signed by an external commercial CA . . . 124
Example 7 for RACDCERT: Setting up secure connections using certificates signed by an internal CA . . . 130
Example 8 for RACDCERT: Setting up client authentication using client certificates signed by an internal CA . . . 135
Example 9 for RACDCERT: Setting up client authentication using client certificates signed by an external CA . . . 140
Example 10A: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1 Release 4 or later) . . . 145
Example 10: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1 Release 3 or earlier) . . . 151
Related information . . . 157
Related documentation and URLs . . . 159
Setting up SSL support for multiple IP addresses 159
Changing the default order of encryption levels used by the Web server . . . 160

Chapter 9. Setting up protection for server resources . . . 161

Before you begin . . . 161
Protection methods . . . 161
Step 1. Activate protection on the server . . . 162
Step 2. Specify which requests you want the server to accept . . . 162
Step 3. Decide which protection options to use . . 163
Options for protecting server resources . . . 163
How the server processes requests . . . 164
Step 4. Create protection setups . . . 165
Directives . . . 165
Subdirectives . . . 166
Creating protection setups for Secure Sockets Layer (SSL) client authentication . . . 168
Step 5. Limit access to individual files . . . 170
Rules for specifying user names, group names, and address templates . . . 171
Group files . . . 172
Use of group files in protection setups . . . 173
Access Control List files . . . 174
Examples. . . . 175
Secure Sockets Layer client authentication hints and tips . . . 176
Default protection scheme when coding %%CLIENT%% on UserID directive . . . 176
Examples: Protecting server resources . . . 177
Example 1: Setting up protection with user names and passwords . . . 177

Example 2: Setting up a protection setup without SSL client authentication 178
Example 3: Setting up a protection setup with SSL client authentication. 181
Changing expired passwords 182

Part 5. Advanced Configuration 183

Chapter 10. Customizing cache management with the Fast Response Cache Accelerator 185
Overview. 185
Cache Accelerator eligibility 185
Planning considerations 186
Access to z/OS Workload Management (WLM) 186
Unique subsystem name for WLM 186
TCP/IP stack name 186
Enabling and configuring the Cache Accelerator 186
Monitoring and managing the Cache Accelerator 187
RMF reports. 187
DISPLAY TCPIP command 187

Chapter 11. Customizing logs and reports 189
Tailoring the logs your server keeps 189
Overview of log types 189
Overview of log setup options. 191
Sample scenario for configuring log files . . . 196
Tailoring the reports your server creates 196
Reporting options and considerations 196
Using the HTLOGREP or Web usage mining reporting programs 197
Creating a report template 199
Viewing reports 200
Sample scenarios for configuring reports . . . 200
Using the Web usage mining statistics reports . . 203
Running Web usage mining from the command line. 205
Web usage mining and multiple servers . . . 206
Logging information with System Management Facilities 206

Chapter 12. Customizing your Web site. 209
Using the HTCounter program to display page count, date, time, and text on a Web page 209
Configuration instructions for the HTCounter program 209
Samples 210
Options 211
Using server-side includes to insert information into CGI programs and HTML documents. . . . 214
Considerations for using server-side includes 214
Preparing to use server-side includes 214
Format for server-side includes 215
Directives for server-side includes 215
Using server-side image maps to enable clickable images 221
Syntax. 221
Examples. 222

Chapter 13. Managing your Web server 225
Modes of Operation for the Web server. 225
Standalone Server mode. 226
Scalable Server mode. 226
Running Multiple Servers 227
Workload Management Enablement for the Web server 228
Workload management overview. 228
Configuring workload management to support the Web server 228
How workload management panels are interpreted 233
Configuring your environment for workload management 234
Configuring your application environment name for workload management 234
Example 1: Setting up a single scalable server on your system. 235
Example 2: Setting up multiple scalable servers on your system. 241
Server Activity Monitor 249
Enabling the Server Activity Monitor 249
Accessing statistics and usage information. . . 249
Web server activity statistics 250
Network activity statistics 252
Access log entries 253
Simple Network Management Protocol 253
SNMP commands and protocol 253
Object IDs and variable names for the HTTP Server MIB 254
Defining a management information base file so that you can do queries with variable names . . 269
Simple Network Management Protocol examples: running with a single Web server and a single Simple Network Management Protocol agent 270
Simple Network Management Protocol examples: running multiple Web servers, each with a unique Simple Network Management Protocol agent 273
Creating an e-mail address to receive Simple Network Management Protocol problem reports. 276
Providing a security password for Simple Network Management Protocol 276
Enabling and disabling SNMP support 277
Turning the SNMP support on and off from the IMWHTTPD program 277
System Management Facilities. 277
Turning SMF support on and off from the IMWHTTPD program 278
Turning SMF on and off with the z/OS operator console MODIFY command 278
Controlling the logging of information by SMF 278
SMF record formats 279
z/OS console commands 284

Chapter 14. Rating Web sites and serving rated Web information 285
PICS overview 285
Who can rate Web sites 285
How Web clients use PICS 286
How the HTTP Server helps you manage PICS labels 287
PICS for Web site administrators 287
PICS for rating services and label bureaus . . . 288
How to manage PICS labels from a central file . . 289
Storing the PICS files on your server 290
Managing PICS labels for your own Web site 290
Starting a PICS rating service and label bureau 290
How to create PICS labels 291
PICS label extensions 291
How to request PICS label information 292
How to update the PICS configuration file . . . 293
Using the online Configuration and Administration forms. 293
Editing the PICS configuration file manually 293
Using wildcards in the Platform for Internet Content Selection configuration file 295

Chapter 15. Retrieving Lightweight Directory Access Protocol information 297
Querying the LDAP server 297
LDAP search filters 297
Examples of LDAP search filters 297
Configuring LDAP on the HTTP Server 298
Using LDAP to protect files 300
Using LDAP to share configuration information 300
Determining why the Web server cannot establish a connection to or bind to Lightweight Directory Access Protocol 302

Chapter 16. Running your server as a proxy 303
Web Traffic Express proxy features 303
Definitions 303
Setting up your proxy server 304
Guidelines for the *IBM Web Traffic Express for Multiplatforms User's Guide* 304
Setting up logging for your proxy server . . . 305
Setting up a forward proxy server 305
Setting up a reverse or hidden proxy server . . 306
Turning off your proxy server 307
Understanding the CACHEAGT program 307

Chapter 17. Running your server with multiple IP addresses or virtual hosts . 309
Requirements for using multiple IP addresses or virtual hosts 309
Multiple IP addresses. 309
Virtual hosts. 309
Setting up your Web server to use multiple IP addresses or virtual hosts 310
Welcome pages 310
Mapping rules 310
Access Control 311

Part 6. Programming 313

Chapter 18. Writing Common Gateway Interface programs 315
Overview of the CGI 315
CGI and dynamic documents 316
Uses for CGI 316
FastCGI support 317
Overview. 317
Updates to the Web server configuration file for FastCGI 317
Updates to the FastCGI configuration file . . . 317
Installing Version 2.2 of the FastCGI Developer's Kit 320
Installing Version 2.1 of the FastCGI Developer's Kit 322
Forms and data processing 325
Creating HTML documents that reference CGI programs. 326
Sending information to the server 328
Processing the information 328
Returning output 331
Protecting your programs 331
Environment variables 331
Processing standard search (ISINDEX) documents 331
Passing SSL environment variables to a CGI program 332
Creating CGI source 332
Parsing Routines 332
String Compare Reminders 332
Running CGI programs written in Java™ 333
Response generation 334
CGI Examples 335
CGI C program. 335
CGI REXX program 340
CGI shell script. 342

Chapter 19. Writing GWAPI programs 345
Overview of the GWAPI. 345
General procedure for writing GWAPI programs 346
GWAPI samples 346
Storing GWAPI programs 346
Guidelines for writing GWAPI programs . . . 346
Server request process 347
Application functions. 349
Predefined functions and macros 359
GWAPI configuration directives 360
Compatibility with other APIs. 362
Porting CGI programs 362
GWAPI reference information 363
Authentication and authorization. 363
Environment variables 365
HTCodePage_t, the other data type 366
z/OS GWAPI REXX applications 367
Invoking REXX executable programs as GWAPI applications 367
Writing GWAPI REXX Executable Programs . . 369
GWAPI REXX executable program exit conditions: 373

Example GWAPI REXX service executable program 373
GWAPI REXX downloads 374
Example GWAPI REXX data filter executable program 374
Debugging C/C++ GWAPI programs 375
Overview of support and restrictions 375
Getting started 376
Troubleshooting hints and tips 377

Chapter 20. Accessing LDAP information with the LDAP API 379
Overview of LDAP API 379
Accessing LDAP configuration information . . . 379
Sample LDAP program 380

Part 7. Appendixes 381

Appendix A. Commands. 383
cacheagt command 383
Syntax. 383
Flags 384
Example 384
cgiparse command 384
Syntax. 384
Flags 386
Examples. 386
Exit statuses. 387
cgiutils command 387
Syntax. 387
Flags 387
Examples. 389
htadm command 389
Syntax. 389
Flags 389
Examples. 392
htlogrep command 392
Syntax. 392
Flags 392
Examples. 393
httpd command 393
Syntax. 393
Flags 394
Tracing flags. 396
Signal handling. 399
Examples. 399
IMWHTTPD program 400
Syntax. 400
Parameters 402
Signal handling. 408
Examples. 409
IMWIWM PROC (workload management). . . 409
wwwcmd command 409
Syntax. 410
Parameters 410
Action. 410
z/OS MODIFY console command 410
Syntax. 410
Parameters 411
Tracing parameters 411
Examples. 414
z/OS Workload Management console commands 415
WLM mode 415
Application environments 415
WLM systems 416

Appendix B. Configuration directives 417
Overview of directives 425
Directives that require a stop and restart of the server 425
Directive quick reference 426
Directive syntax guidelines 434
Access control - Set up access control for the server 435
DefProt - Specify default protection setup for requests that match a template 436
Protect - Activate protection setup for requests that match a template 439
Protection - Define a named protection setup within the configuration file 443
Protection subdirectives 444
Basic - Specify required settings 449
BindSpecific - Specify if the server binds to one or all IP addresses. 449
Bounce — Specify the default start option for the sockets setting SO_REUSEADDR 450
DNS-Lookup - Specify whether you want to look up host names of clients 450
HostName - Specify the fully qualified domain name or IP address for the server 451
imbeds - Specify whether server-side includes will be dynamically imbedded 451
InstallPath - Specify an alternate directory installation path 452
NoLastMod - Specify whether LastModified HTTP headers are added to CGI program output. 453
PidFile - Specify the location of the process ID file 453
Port - Specify the port on which you want the server to listen for requests. 454
Recovery — Customize ABEND recovery performed by the Web server 454
ServerRoot - Specify the current working directory of the server 456
ServerToken - Specify whether you want the server to send server identifying information. . 456
SysDumpName - Specify the high-level qualifier of the IEATDUMP data set 456
Userid - Specify the Default Access Control user ID 457
Codepages - Specify default codepage environment 458
DefaultFsCp - Specify server codepage 458
DefaultNetCp - Specify codepage. 459
Directories and Welcome Page - Set viewing options 459
AddBlankIcon - Specify the icon URI used to align the heading of directory listings 460
AddDirIcon - Specify the icon URI for directories on directory listings 460
AddIcon - Bind an icon to a MIME content-type or encoding-type 461

AddParentIcon - Specify the icon URI for a parent directory on directory listings 461
AddUnknownIcon - Specify the icon URI for unknown file types on directory listings . . . 462
AlwaysWelcome - Specify if a welcome file is returned for all directory requests 462
DirAccess - Control directory listings 463
DirReadme - Control directory README files 463
DirShowBrackets - Use brackets around alternative text on directory listings 464
DirShowBytes - Show byte count for small files on directory listings 464
DirShowCase - Use case when sorting files on directory listings 464
DirShowDate - Show date last modified on directory listings 464
DirShowDescription - Show descriptions for files on directory listings 465
DirShowGroup - Show the group ID of files on directory listings 465
DirShowHidden - Show hidden files on directory listings 465
DirShowIcons - Show icons in directory listings 465
DirShowMaxDescrLength - Set the maximum description length on directory listings 466
DirShowMaxLength - Set the maximum length for file names on directory listings 466
DirShowMinLength - Set the minimum length for file names on directory listings 466
DirShowMode - Show file permissions on directory listings 466
DirShowOwner - Show file owner on directory listings 466
DirShowSize - Show file size on directory listings 467
IconPath - Specify the path for the directory listing internal icons 467
Welcome - Specify names of welcome files . . 467
User directories. 469
UserDir - Enable users to have private Web documents 469
Error messages - Customize Web server error messages 469
ErrorPage - Specify a customized message for a particular error condition 470
Log401Error - Specify whether IMW0196I NOT AUTHENTICATED should be written to the error log 476
GWAPI - Specify GWAPI applications for processing 477
Customize Web server process steps. 477
CounterDirectory - Specify a directory for the HTCounter program 484
DebugToolAddr - Identify the workstation running the Remote Debugger. 485
Lightweight Directory Access Protocol - Set up shared configuration for the server 485
LDAPInfo - Define an external LDAP server . . 485
LDAPInfo Subdirectives 486
LDAPInclude - Retrieve configuration file information from the LDAP server 490
Logging and Reporting - Customize logs and generate reports 491
DoReporting — Use logging and reporting options 491
AccessLog - Name the path for the access log file 492
AccessLogArchive - Remove existing access, agent, or referer log files or run a user exit . . 493
AccessLogExcludeURL - Suppress log entries for specific files or directories 494
AccessLogExcludeMethod - Suppress log entries for files or directories requested by a given method 494
AccessLogExcludeMimeType - Suppress log entries for specific MIME types 495
AccessLogExcludeReturnCode - Suppress log entries for specific return codes 495
AccessLogExpire - Remove existing access log files when they reach a given age in days . . . 495
AccessLogSizeLimit - Remove existing access log files when they reach a given collective size . 496
AccessReportDescription - Give a short description of the HTLOGREP report to be created 496
AccessReportDoDnsLookup- Display client hostnames in HTLOGREP access reports . . . 497
AccessReportExcludeURL - Suppress log entries for specific files or directories from the HTLOGREP report 497
AccessReportIncludeURL - Include only log entries for specific files or directories in the HTLOGREP report 498
AccessReportExcludeHostName - Suppress the log entries for specific host names from the HTLOGREP report 498
AccessReportIncludeHostName - Include only log entries for specific host names in the HTLOGREP report 499
AccessReportExcludeMethod - Suppress the log entries of a given method type from the HTLOGREP report 499
AccessReportExcludeReturnCode - Suppress the log entries with a given return code from the HTLOGREP report 500
AccessReportRoot - Name the path for the root directory where HTLOGREP access log reports are stored 500
AccessReportTemplate - Name the HTLOGREP report template. 501
AccessReportTopList - Specify the top number of items on which HTLOGREP is to report . . 501
AgentLog - Name the path for the agent log file 502
CacheAccessLog - Specify the path for the cache access log files 502
CgiErrorLog - Name the path for the CGI error log file 503
ErrorLog - Name the file where you want to log internal server errors 503
ErrorLogArchive - Remove existing error or CGI error log files or run a user exit 504

ErrorLogExpire - Remove existing error log files when they reach a given age in days 505
ErrorLogSizeLimit - Remove existing error log files when they reach a given collective size . . 505
LogFormat - Specify common or old log file format. 506
LoggingReportingDebugOutput— Generate debug log for HTLOGREP reporting program . 506
LoggingReportingProgram — Specify the reporting program to be used 509
LoggingReportingProgramOptions — Specify reporting program options 509
LogTime - Specify GMT or local time stamps in log files 510
LogToSyslog - Log access information to the syslog, in addition to or instead of the error log . 510
NoLog - Suppress log entries for specific hosts or domains matching a template 511
ProxyAccessLog - Name the path for the proxy access log file 511
RefererLog - Name the path for the referer log file 512
ReportDataCompressionProgram - Specify path to the compression program 512
ReportDataUnCompressionProgram - Specify path to the uncompression program 513
ReportDataCompressionSuffix - Specify the suffix appended to the compressed report data files 513
ReportProcessOldLogs - Check for old logs in the log directory 513
ReportDataSizeLimit - Remove existing access data files when they reach a given collective size. 514
ReportDataArchive - Specify whether to remove existing accessdata files 514
ReportDataExpire - Remove existing access data files when they reach a given age in days . . . 515
SMF - Specify the type of information that SMF records 516
SMFRecordingInterval - Specify how often to record performance record information 516
Meta-Information - Name meta-information files and directories 517
MetaDir - Specify name of subdirectory for meta-information files 517
MetaSuffix - Specify the suffix for meta-information files 518
Methods - Set method acceptance 518
Disable - Disable HTTP methods 519
Enable - Enable HTTP methods 519
Multi-format processing - Define file extensions for multi-format processing 520
Multi-Format Processing. 520
AddLanguage - Specify the language of files with particular suffixes 522
AddEncoding - Specify the MIME content encoding of files with particular suffixes . . . 523
AddCharSet - Specify the character set documents are encoded in 523
AddType - Specify the data type of files with particular suffixes 523
AddClient - Specify file extensions for requesting clients 527
SuffixCaseSense - Specify whether suffix definitions are case sensitive 529
Proxy server settings — Configure the Web server as a proxy server 529
CacheDefaultExpiry - Specify a default expiration date 529
CacheExpiryCheck - Specify whether the server will return expired files 530
CacheLastModifiedFactor - Specify fraction of Last-Modified date to use in determining expiration date 530
CacheLimit_2 - Specify upper limit for cached file size 530
CacheNoConnect - Specify stand alone cache mode 531
CacheOnly - Cache only files with URIs that match a template 531
CacheRoot - Specify cache root directory . . . 531
CacheSize - Specify cache size 532
CacheUnused - Specify how long to keep unused cached files that match a template. . . 532
Caching - Turn proxy caching off or on. . . . 533
ftp_proxy - Specify a proxy server for this proxy to connect to for FTP requests 533
Gc - Turn garbage collection on or off 533
GcDailyGc - Specify a daily time for garbage collection 534
GcMemUsage - Specify how much memory to use for garbage collection 534
gopher_proxy - Specify a proxy server for this proxy to connect to for Gopher requests . . . 534
http_proxy - Specify a proxy server for this proxy to connect to for HTTP requests 535
no_proxy - Connect directly to domains matching templates 535
NoCaching - Do not cache files with URIs that match a template 535
ProxyAccessLog — Name the path for the proxy access log file 536
SocksServer - Specify a socks server 536
Resource mapping - Redirect URLs 536
Exec - Run a CGI program for matching requests 537
ExecDirPass - Control access to directories matching Exec directives 539
Fail - Reject matching requests 539
Map - Change matching requests to a new result string 540
Pass - Accept matching requests 541
Redirect - Send matching requests to another server 543
InheritEnv - Specify which environment variables are inherited by CGI programs . . . 544
DisInheritEnv - Specify which environment variables are disinherited by CGI programs . . 545
Security - Set up secure connections for the server 545

KeyFile - Specify the name of a key database or SAF key ring to be used for SSL connections . . 546
NormalMode - Turn port on or off for HTTP connections 546
SSLCipherSpec - Specify the levels of encryption to use for SSL connections 547
SSLClientAuth - Select the type of SSL client authentication 549
SSLMode - Turn SSL on or off 550
SSLPort - Set port for SSL security 550
SSLServerCert - Associate a key database with an IP address 551
SSLV2Timeout — Set SSL V2 session timeout value 552
SSLV3Timeout — Set SSL V3 session timeout value 552
SSLX500CARoots — Specify location of trusted CA certificates 553
SSLX500Host — Specify host name or IP address of the X.500 directory server 553
SSLX500Port — Specify X.500 directory server port number. 554
SSLX500UserID — Specify user ID for the LDAP connection to the X.500 directory server . 554
SSLX500Password — Specify user ID password for the LDAP connection to the X.500 directory server 555
System Management - Define system management settings 555
ApplEnv - Specify application environment for workload management 555
ApplEnvConfig - Specify a set of directives to tailor the application environment 556
ApplEnvMax - Specify maximum number of Queue Servers that WLM is to maintain for a specific application environment 558
ApplEnvMin - Specify minimum number of Queue Servers that WLM is to maintain for a specific application environment 558
ApplEnvPrestart - Specify an application environment to start at initialization. 559
PluginDefault - Specify default Plugin action 560
PluginExclude - Specify Plugin not to load during initialization 560
PluginInclude - Specify Plugin to be loaded during initialization 561
SNMP - Enable and disable SNMP support . . 563
SNMPCommunityName - Provide a security password for SNMP 564
WebMasterEmail - Create an e-mail address to receive SNMP problem reports 564
Timeouts - Close connections automatically . . . 564
InputTimeout - Specify time allowed for the client to send a request 564
OutputTimeout - Specify maximum time for sending output to the client 565
ScriptTimeout - Specify time allowed for a program to complete 565
Tuning - Define performance and scalability settings 566
CacheLocalFile - Specify files you want to load in memory at start up 566
CacheLocalMaxBytes - Specify maximum amount of memory to use for file caching . . . 567
CacheLocalMaxFiles - Specify the maximum number of files for caching 567
EnableFRCA — Turn dynamic caching on or off 567
Enclave — Create and join a Workload Management enclave for the current request . . 568
FRCAAccessLog — Specify a log file path and name for dynamic caching requests 569
FRCACacheEntries — Specify the maximum number of files to be dynamically cached . . . 570
FRCACacheOnly — Specify URIs to be dynamically cached 570
FRCACacheSize — Specify size of the dynamic cache 570
FRCAMaxFileSize — Specify maximum file size for the dynamic cache 571
FRCANoCaching — Exclude URIs from the dynamic cache 571
FRCAStackName — Specify the TCP/IP stack that supports the dynamic cache 571
FRCAVirtualHost — Indicate to the dynamic cache whether multiple virtual hosts or IP addresses are being used 572
FRCAWLMParms — Specify parameters for Workload Management 572
ListenBacklog - Specify the number of listen backlog client connections for the server to carry 573
LiveLocalCache - Specify whether the cache is updated when a cached file is modified . . . 573
MaxActiveThreads - Specify the maximum number of active threads 574
MaxContentLengthBuffer - Set the size of the buffer when computing content length 575
MaxPersistRequest - Specify the maximum number of requests to receive on a persistent connection 576
PersistTimeout - Specify time to wait for the client to send another request 576
QOS - Specify Quality of Service classification information you want to send to TCP/IP on each request. 577
ServerPriority - Specify the priority you want your server to have on your system 577
UseACLs - Specify whether ACL files will be checked 577
UseMetaFiles - Specify whether meta files will be used 578
WLMSN- Enable the creation of request level Workload Management enclaves 578

Appendix C. Data set naming reference 581

Appendix D. Environment variables 583

Overview. 583
Variables with values that are read-only 584
Variables with values that can be set or created . . 589

Server-Side Include variables 595

Appendix E. GWAPI MVSDS DLL Service 597
Preparing the MVSDS DLL Configuration File . . 597
Specifying the MVSDS DLL configuration directives 597
Enable MVSDS DLL preloading of z/OS data sets using GWAPI directives in the server configuration file 598
Accessing Web contents from z/OS data sets 599
Performance issues when accessing z/OS data sets 600
Return codes from MVSDS functions 601

Appendix F. Messages 603
z/OS LookAt online message facility 603
Support services and resources 604
Message catalog errors 604
Overview of IMW Messages 604
Message severity 604
Message ID ranges and types 604
IMW0001E-0572E: IMWHTTPD Messages 605
Explanation of errno and errno2 codes in messages 605
Message descriptions 605
IMW2000E-2026E: Proxy Server Messages 658
IMW3501I-3546E: CONSOLE Messages 661
Explanation of process descriptor in messages 661
Message descriptions 661
IMW3701E-3730E: HTCounter Program Messages 666
IMW4000E-4018E: HTIMAGE Messages 669
IMW5001E-5010E: HTADM Messages 670
IMW6102I-6805E: SSL Security Messages 672

Appendix G. TCP/IP reference 675
Selecting files or data sets 675
Data set search order 675
z/OS UNIX System Services data set environment 675
TCP/IP file placement configuration 676

Glossary 677

Bibliography 679
IBM HTTP Server 679
WebSphere Application Server 679
z/OS 679

Notices 681
Trademarks 682

Index 685

Summary of Changes for IBM HTTP Server Version 5.3 for z/OS

SC34–4826–03 for z/OS Version 1 Release 4

This book is available in softcopy format only. The most current version is available in a Portable Document Format (PDF) file on the Web at uniform resource locator (URL):

`http://www.ibm.com/software/webservers/httpservers/doc53.html`

To ensure that you have the current version, check this Web page for updates. You can also access the WebSphere Application Server documentation and other related documentation from this Web page.

A vertical line indicates technical changes or additions to the left of the change.

Security

Secure Sockets Layer (SSL) examples that use the new gskkyman utility

The menus for the gskkyman utility change beginning with z/OS Version 1 Release 4. The documentation adds a set of Secure Sockets Layer (SSL) examples that step the customer through these new menus. For more information, see "Examples: Setting up secure connections" on page 69.

Certificate file types generated by the gskkyman utility

New documentation discusses certificate file types for certificates that the gskkyman utility generates in z/OS Version 1 Release 4 and later releases. For more information, see "Certificate file types generated by the gskkyman utility" on page 157.

Hardware encryption

New documentation for z/OS Version 1 Release 4 and later releases explains how to determine if you use hardware encryption for Web server encryption. For more information, see "Hardware encryption" on page 65.

Messages

Changed

Updates for the following message are in Appendix F, "Messages" on page 603:

- IMW6802E

Welcome!

The HTTP Server is a scalable, high-performance Web server that brings you state-of-the-art security, dynamic caching capabilities, advanced server statistic reporting, and site indexing. It allows you to exploit Java™ to build dynamic, personalized Web sites and use the Platform for Internet Content Selection (PICS) to both rate and filter Web content. With the HTTP Server, you can establish an effective presence on the World Wide Web, reach customers and suppliers around the world, and conduct secure electronic commerce.

For the latest HTTP Server product offerings, information, and news, visit the Web site at URL:

`http://www.ibm.com/software/webservers/httpservers/`

Product information

The HTTP Server provides product documentation in softcopy formats only. The most current information is available on the Web site Library page at URL:

`http://www.ibm.com/software/webservers/httpservers/doc53.html`

For a summary of available z/OS documentation and online information, see the *z/OS Information Roadmap*. You can access z/OS documentation on the Web at URL:

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

Support services

For information on support options and resources, refer to the Support section of the Web-based *WebSphere Troubleshooter for z/OS and OS/390*.

To link to the *Troubleshooter*, go to URL:

`http://www.ibm.com/software/webservers/httpservers/troubleshooter.html`

Part 1. Planning

Chapter 1. Planning for installation

Requirements 3
APARs and service updates 3
Compatible and tested browser levels 3
Tested browsers 3
Web server Configuration and Administration Forms 4
IBM Software Development Kit 5
z/OS UNIX System Services 5
Authorization environment considerations . . 5
z/OS UNIX SAF classes used by the Web server. 5
z/OS UNIX System Services user ID 6
DLL considerations within the z/OS UNIX environment 6
WebSphere Application Server 7
Considerations. 7
ABEND recovery performed by the Web server. . 7
Access control with z/OS RACF or other security products. 8
Performance considerations for RACF and SAF-based security products 8
Fast Response Cache Accelerator 8
MVSDS DLL z/OS data set considerations . . . 9
Performance 9
z/OS UNIX System Services BPXPRMxx recommendations. 9
Web server MaxActiveThreads setting . . . 10
Tuning hints and tips on the Web 10
Related performance information and considerations 10
Reporting programs used by the Web server . . 10
Security 10
Migration considerations 10
Security examples 10
Environment updates for gskkyman 10
Key rings defined in a SAF security product 11
SSL and hardware encryption 11
Workload Management 11

Requirements

APARs and service updates

For the most current information on APAR fixes and service updates, check the *z/OS Program Directory*, the product Preventive Service Planning (PSP) documentation, and IBMLink.

You can access the PSP documentation by going to the Support or Documentation section of the Web-based *WebSphere Troubleshooter for z/OS and OS/390*.

To link to the *Troubleshooter*, go to URL:

`http://www.ibm.com/software/webservers/httpservers/troubleshooter.html`

PSP documentation is also available on IBMLink. To access IBMLink on the Web, go to URL http://www.ibm.com/ibmlink/.

Compatible and tested browser levels

To configure the Web server using the Configuration and Administration Forms, you need a browser that:

- Can display frames
- Supports Java™ Development Kit (JDK) 1.1.x
- Is enabled for both Java™script and Java™
- Has color resolution set to at least 256 colors (operating system setting)
- Is set to cache documents and compare the cached document with the network document **every time**

Tested browsers

Note: For the most current information on browser requirements, tested browsers, and browser hints and tips for the Web server, see the Browsers section of the Web-based *WebSphere Troubleshooter for z/OS and OS/390*.

To link to the *Troubleshooter*, go to URL:
`http://www.ibm.com/software/webservers/httpservers/troubleshooter.html`

Web server Configuration and Administration Forms

Table 1. Browsers tested with the Web server Configuration and Administration Forms

Operating system	Browser level
Windows NT **(1)** and Windows 95	• Netscape: – Navigator 4.08 and 4.04 – Navigator Gold 3.01 – Communicator 4.51, 4.5, and 4.04 The JDK 1.1 SmartUpdate patch is required. **(2)** • Microsoft Internet Explorer 5, Version 5.00.2014.0216IC **(3)** To verify the version number, click Help, then About Internet Explorer. • Microsoft Internet Explorer Version 4.0, upgrades: – V4.71.1712.6 – V4.72.2106.8 To verify the upgrade number, click Help, then About Internet Explorer.
AIX	• Netscape Navigator V4.07 and V4.04
OS/2	• Netscape Navigator V4.04 with JDK 1.1.7 **(4)** • Netscape Navigator V2.02 with service level 7 (only the runtime code is required)**(5)**

Hints and tips:

- Help is launched in its own separate window. At times, you might need to move the help window to see the original window under it.
- In Netscape Navigator, when you maximize the configuration browser window and then resize it, you might lose the **help** and **restart** icons. Simply maximize the window to make them reappear.

Notes:

(1): Windows NT 4.0 should be configured to run with **more than** 256 colors to enable GIFs to display properly.

(2): The browser and JDK 1.1 SmartUpdate patch are two **separate installations**. The JDK 1.1 SmartUpdate patch updates the JDK level to 1.1.4.

To confirm that you have the correct level, look at the header in the Netscape Java™ Console (available from the Netscape Communicator menu). The header should say "Netscape Communications Corporation -- Java 1.1.4."

To download the JDK 1.1 SmartUpdate patch, go to URL http://www.netscape.com/.

(3): When using the Configuration and Administration Forms, note that Microsoft Internet Explorer 5, Version 5.00.2014.0216IC may prompt you for your administrative ID and password each time you perform a task. This is a browser problem.

(4): You can download Netscape Navigator V4.04 and the JDK 1.1.7 from the IBM Software Web site at URL http://service.boulder.ibm.com/asd-bin/doc/.

(5): To confirm that you have Netscape Navigator service level 7, open the Installation utility in the Netscape folder. Select the Netscape 2.02.00 item and click Details, then Product Status. Select Netscape Navigator then click Service Level.

To confirm that you have Java 1.1.2 or higher, enter **java -version** on the command line.

If you have not installed Netscape Navigator, install Java first. When you install Navigator, you will be prompted for the Java level.

If you have already installed Navigator, install Java and then click Java™ Version Selection in the Netscape folder to select Java 1.1.

You can download Navigator and Java from the IBM Software Web site at URL http://service.boulder.ibm.com/asd-bin/doc/. If you are installing Java 1.1.4, you need OS/2 Feature Installer Version 1.1 or higher. You can download the latest version of OS/2 Feature Installer from this site.

IBM Software Development Kit

If you plan to install WebSphere Application Server or use Java™ CGIs on the Web server, you must install the correct version of the IBM Software Development Kit (SDK), depending on your version of the Application Server. You can download the version of the SDK that you require from the Java for z/OS Web site at URL:

`http://www.ibm.com/s390/java/`

For WebSphere Application Server requirements for the SDK, refer to the Application Server documentation on the Web at URL:

`http://www.ibm.com/software/webservers/appserv/library_390.html`

z/OS UNIX System Services

Authorization environment considerations

Set up user ID authorizations and security authorizations before starting the Web server. For detailed information on setting up users and security considerations, see the *z/OS UNIX System Services Planning* book. To access this book on the Web, go to the z/OS Book Server Web site at URL:
`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

z/OS UNIX SAF classes used by the Web server

In a UNIX system it is common for one person to have full administrative authority. In a z/OS system, it is common for these administrative authorities to be divided among several people.

The Web server uses the following z/OS UNIX System Authorization Facility (SAF) classes:

BPX.DAEMON
: The Web server usually uses this facility for daemon programs that need to validate user passwords and then change the MVS identity and z/OS UNIX UID and GID of a spawned address space.

BPX.SERVER
The Web server usually uses this facility for its programs that use POSIX threads and need to associate a Surrogate MVS identity with each thread in their address space.

BPX.SMF
The Web server usually uses this facility to validate read access to the its user ID for writing SMF records.

z/OS UNIX System Services user ID

A z/OS UNIX user ID (UID) executes the Web server and validates incoming requests. This book and the sample files shipped in **/***install_path***/samples** assume you have chosen WEBSRV for the z/OS UNIX UID. The *install_path* is the root directory of your Web server installation; the default install path is **/usr/lpp/internet**.

Notes:

1. You must set WEBSRV up as a Web server surrogate user ID in "Completing and customizing your Web server installation" on page 20.
2. The example shows WEBSRV defined with a UID of 0 so that it will always run with superuser authority. The Web server will run successfully whether the Web server UID is 0 or nonzero. To define a Web server UID as nonzero, follow the instructions in "Defining the Web server with a nonzero user ID" on page 21.

   The Web server changes to either a surrogate user ID or the client's local z/OS user ID prior to accessing the requested resource, unless the USERID directive is coded with %%SERVER%%. In this case, the Web server uses its own user ID to access the data.
3. If you have defined the BPX.DAEMON facility class, WEBSRV must be given READ access. Also, you must turn on program control and indicate that the server and Language Environment LOADLIBs are trusted programs.
4. If you have defined the BPX.SERVER facility class, WEBSRV must be given UPDATE access. You must also turn on program control and indicate that the server and Language Environment LOADLIBs are trusted programs.

DLL considerations within the z/OS UNIX environment

This section applies only if you are creating your own dynamic link libraries (DLLs).

A DLL is a file containing executable code and data bound to a program at load time or run time. Some DLL considerations:

- DLLs generated for a BPX.DAEMON environment must be loaded from a program control protected data set or HFS file to ensure the DLL does not corrupt the HTTP Server address space.
- The DLL must adhere to z/OS naming conventions.

For HFS resident DLLs, add the DLL path to the LIBPATH statement in the Web server httpd.envvars file. If the DLL is in an MVS dataset, you must define an external link. For example, to issue an external link for MVS dataset DLLs defined from the z/OS UNIX shell, issue the following:

ln -e IMWDFTFM /usr/lpp/internet/bin/*urdll.so*

Notes:

1. The **ln** command is case sensitive.

2. For *urdll.so*, enter the DLL to be loaded from the server. Since *urdll.so* does not adhere to z/OS naming conventions (namely, "." is not a valid character in a PDS member name and not eight characters or less), an external link to the load module IMWDFTFM is placed in the**/usr/lpp/internet/bin/** directory.
3. The member IMWDFTFM is located in the z/OS search order when running from the shell.

For more information, see the *z/OS UNIX System Services Command Reference*. Find additional information about DLLs in the *z/OS C/C++ Programming Guide* book.

The data sets containing the DLLs should be included in the LPA, LNKLST, or STEPLIB. For performance, LPA is the preferred location, then LINKLST, then STEPLIB.

Alternately, a DLL file stored in an HFS can be used if it is marked program controlled with the **extattr +p efn** command. You need READ access to the BPX.FILEATTR.PROGCTL facility. For more information about the **extattr** command, see the *z/OS UNIX System Services Planning* book.

WebSphere Application Server

WebSphere Application Server for z/OS provides a foundation for delivering the latest enterprise Java™ technologies to support e-business application development and deployment. Using WebSphere Application Server, you can develop and deploy new high-volume transaction e-business applications and Web-enable legacy applications and databases on z/OS.

For planning, installation, and configuration information, refer to the Application Server documentation. You can access the latest Application Server documentation on the Web at URL:

`http://www.ibm.com/software/webservers/appserv/library_390.html`

Considerations

ABEND recovery performed by the Web server

The Recovery directive enables you to customize how the Web server will handle error recovery when an ABEND occurs.

By default, the Web server will attempt to recover from an ABEND and will take the following actions in response to a recoverable situation:

- Issue messages to the MVS console, trace log, and error log
- Take a formatted CEEDUMP of the environment at the time of the ABEND
- Take an unformatted IEATDUMP of the environment at the time of the ABEND. You must set the SysDumpName directive for the IEATDUMP to be generated.
- Continue processing

To properly obtain the CEE dump, you must have a CEEDUMP DD statement in your JCL if you are starting the Web server from a PROC. If you are starting the Web server from the z/OS UNIX shell, ensure you have the _CEE_DMPTARG environment variable set to the path where you want the dump to be stored.

For information on error recovery options, see "Recovery — Customize ABEND recovery performed by the Web server" on page 454.

For information on the SysDumpName directive, see “SysDumpName - Specify the high-level qualifier of the IEATDUMP data set” on page 456.

Access control with z/OS RACF or other security products

For controlling access to Web resources on MVS, the Web server provides extensive access control support, including system validation of user IDs and passwords and access control through surrogate user ID support.

Resource Access Control Facility (RACF), or an equivalent security product, manages system and data security by verifying a user’s identity and access to a resource.

In general, System Authorization Facility (SAF) security checking with RACF, or another vendor’s security product, is recommended because it can protect the installation from unauthorized access to MVS as well as Web server resources.

Users are identified by an z/OS UNIX user ID (alphanumeric) kept in the RACF user profile, and an z/OS UNIX group ID (GID) kept in the RACF group profile. For more information on setting up user and group IDs, see the *z/OS UNIX System Services Planning* book.

Performance considerations for RACF and SAF-based security products

If using RACF or another SAF-based security product, consider the following hints and tips:

- Use surrogate user IDs, if that is acceptable from a security point of view. For more information on setting up surrogate user IDs, see “Web server surrogate user IDs” on page 22.
- If you use %%SAF%% with the PUBLIC surrogate user ID, this provides protection of your server resources without a high CPU cost. OS/390 Release 3 or greater automatically enhances performance when you use surrogate user IDs and RACF.

  If you require Web clients to have a unique SAF-based user ID and password on the Web server system, specifying %%SAF%% with %%CLIENT%% will give you more robust security but at a higher CPU cost.
- Put your RACF data set on a control unit with caching and the DASD fast write feature. Instructions for enabling caching are in the documentation for DFSMS. Contact your DASD support programmer.
- Cache highly used pages.
- Clean up the RACF database periodically.
- Ensure that you have turned on program control for all required DLLs and optional functions such as WLM or SMF. For more information, see the program control requirements in “Completing and customizing your Web server installation” on page 20.

For more information on RACF, see *z/OS Security Server (RACF): Security Administrator’s Guide*.

Fast Response Cache Accelerator

The Fast Response Cache Accelerator can improve the performance of the HTTP Server when serving text and image files over a non-secure connection. Dynamic content and protected pages are not cached.

For more information, see Chapter 10, “Customizing cache management with the Fast Response Cache Accelerator” on page 185.

MVSDS DLL z/OS data set considerations

The Web server can use a DLL named MVSDS to preload a z/OS data sets. Preloading of a z/OS datasets is suggested for frequently accessed Web content. Note that MVS data sets can be accessed without preloading them.

Decide which, if any, data sets are to be preloaded when the server is started. Then add these datasets to the MVSDS DLL configuration file and specify the MVSDS DLL configuration file name on the ServInit directive in the server configuration file. The default MVSDS DLL configuration file is `/etc/mvsds.conf`.

Related Information:
- Appendix E, “GWAPI MVSDS DLL Service” on page 597
- “ServerInit - Customize the initialization step” on page 477

Performance

Note:
This section includes tuning hints and tips that should improve Web server performance. Information has been gathered under testing conditions or from Web server users. Results in your environment may vary. In addition to the hints and tips in this section, check the Preventive Service Planning (PSP) bucket for the latest recommended maintenance and information on PTFs that address performance issues.

z/OS UNIX System Services BPXPRMxx recommendations

In tuning z/OS UNIX, options should be set in the BPXPRMxx concatenation specified by the OMVS parm of the IEASYSxx parmlib member. Most can also be set by the /setomvs command; settings can be checked using the /d omvs,o command. If settings are modified with /setomvs commands, the Web server must be restarted to pick up the new values.

Recommended settings:

MAXTHREADTASKS
Set between 1000 and 5000.

MAXTHREADS
Set to twice the value of MAXTHREADTASKS.

MAXFILEPROC
A setting of 10000 is usually adequate for about 150 concurrent users. At higher request rates, a good approximation is the number of Web requests per second multiplied by 120 up to 65535. A setting of 65535 should be used only for extremely high volume Web throughput, for example, thousands of requests per second.

MAXSOCKETS
Set at least as high as MAXFILEPROC. There is less of a performance penalty if this value is too high.

IPCSHMMPAGES
The default of 256 should be sufficient when running the Web server in Standalone mode. If the Web server is running in Scalable Server mode, increase this value to 12800.

Web server MaxActiveThreads setting

The MaxActiveThreads directive sets the maximum number of threads that you want to have active at one time. If the maximum is reached, the server holds new requests until another request finishes and threads become available.

Recommended setting: If the Web server is running in Scalable Server mode, a value of 100 or less is recommended; for Standalone mode, a value of 150 or less is recommended. If the percentage of static pages served is high, for example 60% or more, a higher setting may be needed. Experience has shown that a value greater than 200 can cause storage shortages. The setting of MaxActiveThreads must be lower than the z/OS UNIX BPXPRMxx setting for MAXTHREADTASKS.

Tuning hints and tips on the Web

For the most current tuning information, see the WebSphere Troubleshooter for z/OS tuning hints and tips at URL:

`http://www.ibm.com/software/webservers/httpservers/doc/v51/ttunc.htm`

Related performance information and considerations

- "Performance considerations for RACF and SAF-based security products" on page 8
- "Performance considerations when starting the Web server" on page 39
- "Tuning - Define performance and scalability settings" on page 566
- "System Management - Define system management settings" on page 555
- Chapter 10, "Customizing cache management with the Fast Response Cache Accelerator" on page 185
- "Workload Management Enablement for the Web server" on page 228
- "Server Activity Monitor" on page 249
- "System Management Facilities" on page 277

Reporting programs used by the Web server

You can use the default IBM reporting program, HTLOGREP, or specify a third-party reporting program. For more information, see "Tailoring the reports your server creates" on page 196.

Security

Migration considerations

If you are migrating from a previous release of the Web server, review the following sections:

- "Certificate and key management changes" on page 15
- "Certificate authority utility changes" on page 16

Security examples

Before setting up security, review the examples in "Examples: Setting up secure connections" on page 69.

Environment updates for gskkyman

In Version 5.3, IKEYMAN has been replaced by the z/OS System SSL gskkyman utility. System SSL gives you the option of using either the shell-based gskkyman program or the z/OS RACF RACDCERT command for certificate and key management tasks. If you use gskkyman, ensure that your system environment is set up to support System SSL and the gskkyman utility. For information on setup requirements, see the *z/OS System SSL Programming Guide and Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Key rings defined in a SAF security product

If you create and define key rings using a System Authorization Facility (SAF) security product such as RACF, you must give the Web server ID access to the key rings and specify the SAF option on the KeyFile directive. For more information, see "Step 3. Enable the Web server to use optional functions" on page 25.

SSL and hardware encryption

V5.0 of the Web server added new SSL enhancements that allow the installation to designate a preference order for cipher specifications using the SSLCipherSpec directive. This enables you to configure the Web server to perform DES and Triple-DES encryption using hardware encryption. For information on enabling this support on the Web server, see "Hardware encryption" on page 65.

Workload Management

In a z/OS production environment, Workload Management (WLM) running in Goal mode can be used to balance workloads and distribute resources among competing workloads. To exploit the benefits of WLM, the Web server can be enabled for WLM support, that is, running in Scalable Server mode. This means that the Web server is configured for WLM support using the ApplEnv directive and is started using the -SN (subsystem name) parameter. The ApplEnv directive is used by the Web server to divide incoming requests into Application Environments and to route those requests.

You should know about the following tasks when considering using workload management:

- Formatting coupled data sets
- Permitting workload management to RACF
- Using workload management panels
- Using operator console commands to verify workload management is working

Related Information:

- For information on permitting WLM to RACF, see "z/OS Workload Management" on page 27.
- For overview and configuration information, see "Workload Management Enablement for the Web server" on page 228.
- For a description of WLM configuration directives, see "System Management - Define system management settings" on page 555.
- For additional information on WLM, see *z/OS MVS Planning: Workload Management*.

Chapter 2. Migration considerations

Installation 13
Backing up existing configuration files 13
Using Version 5.3 configuration files 13
Removing IMW.SIMWMOD1 from your LNKLST and start procedure. 13
Reviewing Version 5.3 installation and customization information 13
IMWHTTPD program 14
Java™ support 15
Proxy server and caching proxy server 15
Security 15
Certificate and key management changes . . . 15
Certificate authority utility changes 16

Installation

Backing up existing configuration files

To complete your Web server install, you run the setup.sh program. If you install the server in the default install path (/usr/lpp/internet), setup.sh copies existing configuration files to the /usr/lpp/internet/etc and /etc directories. If you specify an install path other than the default, setup.sh copies existing configuration files to *install_path*/etc.

If your existing configuration files are located in the default server install directory and use the default file names, we recommend that you copy those files into a new directory before running setup.sh.

Using Version 5.3 configuration files

Make sure you are pointing to copies of the httpd.conf and httpd.envvars files that are shipped with Version 5.3 of the Web server. We recommend that you start your new Web server with copies of the sample httpd.conf and httpd.envvars files shipped with Version 5.3 before manually adding changes from your previous release files. Updating old levels of these files to use with the new Web server can cause problems and errors.

Removing IMW.SIMWMOD1 from your LNKLST and start procedure

Prior to Version 5.1, the Web server program (IMWHTTPD) was installed in IMW.SIMWMOD1. For Version 5.1 and later, the IMWHTTPD program is installed in SYS1.LINKLIB.

Because of this change, you must remove the IMW.SIMWMOD1 load library from your LNKLST concatenation and remove the STEPLIB from your start procedure.

Reviewing Version 5.3 installation and customization information

If you are migrating from a previous release of the Web server, you may have completed many of the steps in “Completing and customizing your Web server installation” on page 20. However, we recommend that you review this information because of updates and changes to the Web server. For example, there are important program control settings that must be defined during Web server customization.

IMWHTTPD program

The following table lists commands (override parameters), the release each command can be used in, and the configuration directive (if one exists) associated with a specific command. For more information, see "IMWHTTPD program" on page 400. For information about configuration directives, see Appendix B, "Configuration directives" on page 417.

In the following table:
- 2.1 and 2.2 refer to the IBM Internet Connection Secure Server.
- 4.6.1 refers to Lotus Domino Go Webserver.
- 5.x refers to Lotus Domino Go Webserver (5.0) and the IBM HTTP Server (5.1 and later).

Table 2. Command, Version, Associated Directive

Command	2.1	2.2	4.6.1/5.x	Directive
-AE	No	Yes	Yes	ApplEnv
-cacheroot	No	No	No	CacheRoot
-db	No	No	No	DirReadme
-debug	No	Yes	Yes	N/A
-dn	No	No	No	DirAccess
-ds	No	No	No	DirAccess
-dy	No	No	No	DirAccess
-dt	No	No	No	DirReadme
-disable	No	No	No	Disable
-enable	No	No	No	Enable
-errlog	No	No	No	ErrorLog
-fscp	Yes	Yes	Yes	DefaultFsCp
-gc_only	Yes	Yes	Yes	N/A
-gmt	No	No	No	LogTime
-h	Yes	Yes	Yes	HostName
-l	No	No	No	AccessLog
-lb	No	Yes	Yes	N/A
-localtime	No	No	No	LogTime
-mtv	Yes	Yes	Yes	N/A
-netcp	Yes	Yes	Yes	DefaultNetCp
-newlog	No	No	No	LogFormat
-nobg	Yes	Yes	Yes	N/A
-nodebug	No	Yes	Yes	N/A
-nodns	No	No	No	DNS-Lookup
-nolog	No	No	No	NoLog
-normalmode	Yes	Yes	Yes	NormalMode
-nosec	Yes	Yes	Yes	N/A
-nosmf	No	Yes	Yes	SMF
-nosnmp	No	Yes	Yes	SNMP
-oldlog	No	No	No	LogFormat

Table 2. Command, Version, Associated Directive (continued)

Command	2.1	2.2	4.6.1/5.x	Directive
-p	Yes	Yes	Yes	Port
-r	Yes	Yes	Yes	N/A
-restart	Yes	Yes	Yes	wwwcmd command
-smf	No	Yes	Yes	SMF
-SN	No	Yes	Yes	Subsys Name
-snmp	No	Yes	Yes	SNMP
-sslmode	Yes	Yes	Yes	SSLMode
-sslport	Yes	Yes	Yes	SSLPort
-v	Yes	Yes	Yes	N/A
-vc	Yes	Yes	Yes	N/A
-version	Yes	Yes	Yes	N/A
-vv	Yes	Yes	Yes	N/A

Java™ support

In previous releases, support for Java™ servlets was provided by ServletExpress and was installed as part of your Web server installation. Beginning in Version 5.1, this support is no longer included with the Web server.

For Java support, you must install WebSphere Application Server. The Application Server provides a Java-based Web application server environment that enables you to develop and deploy new high-volume transaction e-business applications and Web-enable legacy applications and databases on z/OS.

For planning, installation, and configuration information, refer to the Application Server documentation. You can access the latest Application Server documentation on the Web at URL:

`http://www.ibm.com/software/webservers/appserv/library_390.html`

Proxy server and caching proxy server

Beginning in Version 5.0 of the Lotus Domino Go Webserver, the process for defining proxy and caching proxy settings changed.

IBM Web Traffic Express 1.0 provides proxy support for the server and is installed when you install the HTTP Server. For configuration information, see Chapter 16, “Running your server as a proxy” on page 303.

Security

Certificate and key management changes

Prior to Version 5.0 of the Web server, you could create keys and certificate requests using the Configuration and Administration Forms. This option was removed in Version 5.0 for security reasons. In Version 5.0, the IBM Key Management Utility (IKEYMAN) replaced the Make Key File Utility (MKKF) for creating keys and certificate requests. Also in Version 5.0, the IMWSendMail utility was removed and is no longer used for sending certificate requests to a CA.

Migration considerations

In Version 5.3, IKEYMAN has been replaced by the z/OS System SSL gskkyman utility. System SSL gives you the option of using either the shell-based gskkyman program or the z/OS RACF RACDCERT command for certificate and key management tasks. If you define key rings using RACF or another SAF security product, you must permit the Web server to access the key rings and specify the SAF option on the KeyFile directive. For instructions, see "Step 3. Enable the Web server to use optional functions" on page 25.

Related Information:

- For examples, see "Examples: Setting up secure connections" on page 69.
- For migration information, environment setup requirements, and instructions for using gskkyman and the RACDCERT command, see *z/OS System SSL Programming Guide and Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.
- For an overview of security concepts and Web server security options, see Chapter 8, "Setting up a secure server" on page 53.

Certificate authority utility changes

In Version 5.0, the certutil command was replaced by a new server certificate authority utility called Domino Go CA. In Versions 5.1 and 5.2, this utility was called HTTP Server CA.

In Version 5.3, HTTP Server CA has been removed and is no longer available. You can use the z/OS System SSL gskkyman program or the RACF RACDCERT command to set up your own CA, or you can purchase CA software. For more information, see "Acting as your own CA" on page 63.

Part 2. Installing

Chapter 3. Installing your secure server

Before you begin 19
Complete z/OS installation 19
Review Web server planning and migration information 19
Check for information updates on the Web . . . 19
Completing and customizing your Web server installation 20
Step 1. Set up IDs used by the Web server . . . 20
Setting up the the Web server's administration ID 20
Setting up the ID under which the Web server runs 20
Web server access control user IDs 22
Web server surrogate user IDs 22
Step 2. Turn on program control for DLLs . . . 24
Program control for required MVS datasets . . 24
Program control for DLLs stored in the HFS 24
Debugging hints and tips 24
Step 3. Enable the Web server to use optional functions 25
Program control for z/OS System SSL . . . 25
Access to key rings defined in RACF 26
Permitting user IDs to CSFSERV for hardware encryption 26
z/OS Workload Management 27
System Management Facilities 28
Lotus Notes access using RACF user IDs . . 28
Step 4. Make TCP/IP configuration adjustments as needed 28
Step 5. Copy and customize the Web server PROC 29
Step 6. Configure installed files by running setup.sh 30
Before running setup.sh 30
Tasks performed by setup.sh 30
Hints and tips 31
Running setup.sh 31
Step 7. Configure installed files on your target system by running setuptgt.sh (optional) . . . 33
Before running setuptgt.sh 33
Tasks performed by setuptgt.sh 33
Running setuptgt.sh 33
Step 8. Customize your Web server configuration file httpd.conf 35
Using a Web server user ID other than WEBADM 35
Changing default directories for logs and reports 35
Step 9. Customize your Web server environment variables file 35
Step 10. Set up security 36
Step 11. Set up proxy server support (optional) 36
Step 12. Install and configure WebSphere Application Server (optional) 36
What's next? 36

Before you begin

Complete z/OS installation

After completing your z/OS installation, use the instructions in this chapter to complete your installation and customize the Web server.

For information on z/OS installation options and considerations, refer to the *z/OS Program Directory* and the *z/OS Planning for Installation* book. To access these books on the Web, go to the z/OS Book Server Web site at URL:
`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Review Web server planning and migration information

Before you install Version 5.3 of the Web server, review Chapter 1, "Planning for installation" on page 3.

If you are migrating from a previous release of the Web server, also review Chapter 2, "Migration considerations" on page 13.

Check for information updates on the Web

For the most current Web server documentation and updates, go to the Web site Library page at URL:

`http://www.ibm.com/software/webservers/httpservers/doc53.html`

Completing and customizing your Web server installation

Install Path
In the following steps, *install_path* is the root directory of your Web server installation. This value is either the default install path (**/usr/lpp/internet**) or the install path you specify during installation. This path is on the InstallPath directive in your httpd.conf file.

Step 1. Set up IDs used by the Web server

Setting up the the Web server's administration ID

You can configure your Web server by using the Configuration and Administration forms. These forms are a link off the Web server's default Front Page. You will need a z/OS UNIX group ID (GID) and a z/OS UNIXuser ID (UID) to configure and administer the Web server through these forms. This book and the sample files shipped in **/***install_path***/samples** assume you are using IMWEB for the z/OS UNIX GID and WEBADM for the Web server's administration UID.

The following example shows how to define IMWEB and WEBADM using RACF commands. If you are using another security product, refer to that product's documentation for instructions.

```
ADDGROUP IMWEB OMVS(GID(205))
ADDUSER WEBADM DFLTGRP(IMWEB) OMVS(UID(206) HOME('/usr/lpp/internet') PROGRAM('/bin/sh'))
```

Notes:

1. If you use a different GID, you must edit the setup.sh shell script and change all occurrences of IMWEB. If you use a different UID, you must change all occurrences of WEBADM in the Web server configuration file and change the ownership of files in the **/***install_path***/server_root/admin-bin** directory.
2. Members of the IMWEB group have read/write/execute access to all of the files that control the Web server. The Web server sets up this access when you run the setup.sh command in "Step 6. Configure installed files by running setup.sh" on page 30.
3. To use the Web server Configuration and Administration Forms user interface, the WEBADM UID must have read permission to the process ID file (PidFile), read/write permission to the Web server configuration file, and execute permission to the wwwcmd command. The Web server sets up these permissions when you run the setup.sh command in "Step 6. Configure installed files by running setup.sh" on page 30. The PidFile is created when you start the Web server. The default directory and file for the PidFile directive is **/usr/lpp/internet/server_root/httpd.pid**.

Setting up the ID under which the Web server runs

A z/OS UNIX user ID (UID) executes the Web server and validates incoming requests. The Web server executes under this user ID whether you start the server by using a procedure or by using the httpd command. You will assign the user ID to the Web server started task later in the installation.This book and the sample files shipped in **/***install_path***/samples** assume you have chosen WEBSRV for the z/OS UNIX UID.

The following example shows how to define WEBSRV using RACF commands. If you are using another security product, refer to that product's documentation for instructions.

```
ADDUSER WEBSRV DFLTGRP(IMWEB) OMVS(UID(0) HOME('/usr/lpp/internet') PROGRAM('/bin/sh'))
RDEFINE FACILITY BPX.DAEMON UACC(NONE) NOTIFY(WEBSRV)
RDEFINE FACILITY BPX.SERVER UACC(NONE) NOTIFY(WEBSRV)
PERMIT BPX.DAEMON CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
PERMIT BPX.SERVER CLASS(FACILITY) ID(WEBSRV) ACCESS(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
```

Notes:

1. You must set WEBSRV up as a Web server surrogate user ID. For more information, see "Web server surrogate user IDs" on page 22.
2. The example shows WEBSRV defined with a UID of 0 so that it will always run with superuser authority. The Web server will run successfully whether the Web server UID is 0 or nonzero. To define a Web server UID as nonzero, follow the instructions in "Defining the Web server with a nonzero user ID".

   The Web server changes to either a surrogate user ID or the client's local z/OS user ID prior to accessing the requested resource, unless the USERID directive is coded with %%SERVER%%. In this case, the Web server uses its own user ID to access the data.
3. If you have defined the BPX.DAEMON facility class, WEBSRV must be given READ access. Also, you must turn on program control and indicate that the server and Language Environment LOADLIBs are trusted programs.
4. If you have defined the BPX.SERVER facility class, WEBSRV must be given UPDATE access. You must also turn on program control and indicate that the server and Language Environment LOADLIBs are trusted programs.

Defining the Web server with a nonzero user ID: The user ID that executes the Web server and validates incoming requests can either have a UID of 0 (a superuser) or a UID of nonzero. This section describes the tasks that must be done in order to implement a user ID of nonzero.

- **Insure required APAR's are applied**

  In order to enable all Web server capabilities when running with a nonzero user ID, the following APAR's must be applied:

  – Web server APAR PQ41777 for versions 5.0 and above

    Note: PQ41777 is backward compatible. You can therefore continue to run with a UID of 0 without making any of the RACF changes described in this section.

  – Unix System Services APAR OW45077 for OS/390 V2R8 and above.
- **Define nonzero user ID**

  The following example shows how to define JDOE which is a nonzero user ID.

```
ADDUSER JDOE DFLTGRP(IMWEB) OMVS(UID(202) HOME('/u/jdoe')
  PROGRAM('/bin/sh') ASSIZEMAX(2147483647) CPUTIMEMAX(2147483647)
  FILEPROCMAX(065535))
RDEFINE FACILITY BPX.SERVER UACC(NONE) NOTIFY(JDOE)
PERMIT BPX.SERVER CLASS(FACILITY) ID(JDOE) ACCESS(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
```

- **Give the nonzero user ID appropriate permissions to directories and files:** Unix System Services directory and file permissions must be defined so that the nonzero user ID under which the Web server executes can access all appropriate directories and files. For example, you can implement SSL using a key database file and a stash file. The Web server's nonzero user ID must be given read and execute access to the security DLLs, read/write access to the key database file, and read access to the stash file.
- Continue installation of the Web server by first finishing this step and then proceeding to subsequent steps. As you go through these steps, you will issue additional RACF commands for your nonzero user ID where appropriate.

Related information:
- "z/OS UNIX System Services" on page 5

Web server access control user IDs

Web servers frequently need to process requests from users that do not have user IDs on the system running the server. The server uses access control user IDs to access resources for these requests.

The sample configuration provided with the Web server, protects the default Home Page and samples with the special user ID, %%CLIENT%%. This means that only users known to the system can use the Web server, until you explicitly grant access to other users.

%%CLIENT%%
: The Web server requires that the requester have a local z/OS user ID and password. The requester's user ID is used to access the data. The user is prompted for a valid password.

%%SERVER%%
: The Web server uses its own user ID to access data.

Note: Be extremely cautious when using %%SERVER%%. If the Web server is running with a user ID that has superuser authority (UID of 0), requests can be served under superuser authority.

%%CERTIF%%
: The Web server treats SSL connection certificate data in a special way. When presented with an SSL session with client certificate data present, the Web server attempts to map the client certificate to a local MVS User ID and password. The request is treated as if %%CLIENT%% has been specified in the following situations:
- The session is not an SSL session.
- There is no certificate present or the certificate cannot be mapped.
- The underlying support is not available.

Note that SSLCLientAuth must be set on in order to get client certificate data.

Related information:
- "Access control with z/OS RACF or other security products" on page 8
- "Userid - Specify the Default Access Control user ID" on page 457
- "Default protection scheme when coding %%CLIENT%% on UserID directive" on page 176

Web server surrogate user IDs

You probably want to establish several surrogate user IDs with z/OS UNIX access authority appropriate for a group of users or class of requests.

Examples:

In the following list, WEBADM and PUBLIC are the most commonly used surrogate IDs. INTERNAL and PRIVATE IDs may be used but are not required as part of your Web server setup. They are shown here as examples only.

WEBADM
: If you want to use the Web server's remote Configuration and Administration Forms with the supplied sample configuration file, the WEBSRV ID must be allowed to use WEBADM as a surrogate user ID.

WEBSRV is the z/OS UNIX user ID described in "Setting up the ID under which the Web server runs" on page 20.

PUBLIC
This ID has very limited access and is used to handle requests from unknown users on the public network. You might allow these users to view a few informational pages about your company and products. You would not allow these users to store data on your system or execute more than a few well controlled CGI programs.

INTERNAL
This ID has moderate access and is used to handle requests from anyone on your internal corporate network. You might allow these users to view company announcements and standards and perhaps provide a bulletin board application for posting items of general interest.

PRIVATE
This ID has access to a set of restricted information. You might require these users to know a Web administered user ID and password or even to have a valid user ID and password on this system. After validating their user ID and password, you would then access the data under the surrogate user ID PRIVATE.

Example: The following example uses RACF commands to show how to set up surrogate user IDs for a server that executes with the ID WEBSRV. If you have defined your Web server to execute with a nonzero user ID, you can instead set up surrogate IDs for your nonzero user ID. If you are using another security product, refer to that product's documentation for instructions.

1. Activate the SURROGAT class support in RACF by issuing the following command:

   ```
   SETROPTS CLASSACT(SURROGAT)
   ```

   This has to be done only once on the system. The SURROGAT class may already have been set up on your system.

2. Define a SURROGAT class profile for each surrogate user ID:

   ```
   ADDGROUP EXTERNAL OMVS(GID(999))
   ADDGROUP EMPLOYEE OMVS(GID(500))
   ADDGROUP SPECIAL  OMVS(GID(255))
   ADDUSER PUBLIC   DFLTGRP(EXTERNAL) OMVS(UID(998) Home('/') PROG('/bin/sh'))
   ADDUSER INTERNAL DFLTGRP(EMPLOYEE) OMVS(UID(537) Home('/') PROG('/bin/sh'))
   ADDUSER PRIVATE  DFLTGRP(SPECIAL)  OMVS(UID(416) Home('/') PROG('/bin/sh'))
   RDEFINE SURROGAT BPX.SRV.WEBADM   UACC(NONE)
   RDEFINE SURROGAT BPX.SRV.PUBLIC   UACC(NONE)
   RDEFINE SURROGAT BPX.SRV.INTERNAL UACC(NONE)
   RDEFINE SURROGAT BPX.SRV.PRIVATE  UACC(NONE)
   ```

3. Permit the server, WEBSRV, to create security environments using these user IDs:

   ```
   PERMIT BPX.SRV.WEBADM   CLASS(SURROGAT) ID(WEBSRV) ACCESS(READ)
   PERMIT BPX.SRV.PUBLIC   CLASS(SURROGAT) ID(WEBSRV) ACCESS(READ)
   PERMIT BPX.SRV.INTERNAL CLASS(SURROGAT) ID(WEBSRV) ACCESS(READ)
   PERMIT BPX.SRV.PRIVATE  CLASS(SURROGAT) ID(WEBSRV) ACCESS(READ)
   SETROPTS RACLIST(SURROGAT) REFRESH
   ```

Step 2. Turn on program control for DLLs

> **Note:**
> This section includes information on program control requirements for the Web server and known requirements for other products that are frequently used with the Web server. However, if you are using other products with the Web server, ensure that you check their product documentation for any additional program control requirements.

Program control for required MVS datasets

Ensure that program control is turned on for the following MVS datasets:

Note: For *hlq*, enter the high level qualifier for your system installation, for example, SYS1.LINKLIB.

- *hlq*.LINKLIB
- *hlq*.SCEERUN
- *hlq*.SCLBDLL
- If using the Resource Measurement Facility (RMF), *hlq*.SERBLINK
- If using DB2, *hlq*.SDSNLOAD and *hlq*.SDSNEXIT
- If using JDBC, *hlq*.CSSLIB
- If using CICS Transaction Gateway, *hlq*.SDFHEXCI and *hlq*.SDFHLOD1
- If using Tivoli Service Desk, *hlq*.SBLMMOD1(BLGYRXM)
- If using MQSeries, *hlq*.SESQLINK
- If using Run-Time Library Services (RTLS), *hlq*.SCEERTLS
- If you are obtaining an IEATDUMP by setting the SysDumpName directive and setting the Recovery directive to Msg/Dump, Normal, or Full, *hlq*.MIGLIB
- If using C/370, *hlq*.SEDCLINK

The following example shows how to turn on program control using RACF commands. If you are using another security product, refer to that product's documentation for instructions. If you are turning on program control for the first time, you should use RDEFINE statements instead of RALTER statements.

```
RALTER PROGRAM * ADDMEM('hlq.LINKLIB'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('hlq.SCEERUN'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('hlq.SCLBDLL') UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

In this example, an asterisk (*) is used to specify all programs in the data set. You can specify a specific program in the data set.

Program control for DLLs stored in the HFS

Ensure that program control is turned on for the following DLLs:

- Any GWAPI DLL loaded by a service or specified on any GWAPI directives like ServerInit or ServerTerm in the Web server httpd.conf configuration file.
- HFS entries on the Web server LIBPATH environment variable for other products such as DB2 or Net.Data

Debugging hints and tips

If you load a DLL that is not program controlled, you may receive error code 02AF (address space dirty).

MVS datasets

Enhancements to RACF help you more easily determine the cause of

program control authorization failures. Before following the instructions in II08176 or II10548, see your RACF documentation for more information on these enhancements.

DLLs stored in the HFS

Enhancements to RACF help you more easily determine the cause of program control authorization failures. Before following the instructions in II08176 or II10548, see your RACF documentation for more information on these enhancements.

Unix System Services APAR OW44655 adds messages to identify uncontrolled programs. This support is available in OS/390 V2R8 and releases above V2R8. The error message containing a 02af will be printed to standard error. The operator's console will contain a message that indicates the program that was not marked for program control. For example, on standard error, the message would be:

```
[ A1684F8 02/May/2001:07:54:27.033144]: HTAA_ACtl....
  Failed access as Surrogate
: PUBLIC, Errno: 139, Errno2: 0be802af,
  Error: EDC5139I Operation not permitted.
```

Note: In this example each entry appears on two lines for printing purposes. Each entry appears on one line of standard error.

On the operator's console the message would be:

```
BPXP015I HFS PROGRAM /u/user25/regapi/regapi IS NOT MARKED PROGRAM
CONTROLLED.
BPXP014I ENVIRONMENT MUST BE CONTROLLED FOR SERVER (BPX.SERVER)
PROCESSING.
```

Related Information:

- For information on setting _BPX_SHAREAS=NO in the httpd.envvars file to avoid 02AF, address space dirty, see"Step 9. Customize your Web server environment variables file" on page 35.

Step 3. Enable the Web server to use optional functions

Program control for z/OS System SSL

If you set up your Web server to provide secure communications over the Internet, the Web server uses z/OS System SSL to establish the secure connections. Before the Web server can use System SSL, you must:

- Add the System SSL load library, *hlq*.SGSKLOAD, to the linklist.
- Program control *hlq*.SGSKLOAD in RACF.

hlq is the high level qualifier for your system installation, for example, SYS1.SGSKLOAD.

To turn on program control using RACF, issue:

```
RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

If you are turning on program control for the first time, use RDEFINE statements instead of RALTER statements. If you are using another security product, refer to that product's documentation for instructions.

Access to key rings defined in RACF

Terminology Note
In this book, the terms `key database` and `key ring` are similar but not the same. The term `key database` refers to a key database file (*.kdb) you create using the z/OS System SSL gskkyman utility. This key database file contains public keys, private keys, trusted CAs, and certificates. In RACF or other SAF security products, the term `key ring` does not refer to a key database file. A key ring contains public keys, private keys, and certificates, and is stored in a RACF or other SAF database.

You can create keys and certificates using the z/OS System SSL gskkyman utility or a SAF security product such as RACF. If a key ring is created and defined using RACF or another SAF security product, you must:

- Permit the Web server to access the key ring.
- Specify the SAF option on the Web server KeyFile directive.

The examples in this section apply only if you are using RACF. If you are using another security product, refer to that product's documentation for instructions.

The following example shows how to permit the Web server user ID **WEBSRV** to the key ring defined in RACF. If you have defined your Web server to execute with a nonzero user ID, you can instead permit the nonzero user ID to the key ring:

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
PE IRR.DIGTCERT.LIST CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)

RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PE IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)

SETR CLASSACT(FACILITY)
SETR RACLIST(FACILITY) REFRESH
```

For detailed examples on how to set up secure connections using RACF, see "Examples: Setting up secure connections" on page 69.

Permitting user IDs to CSFSERV for hardware encryption

This section assumes that you have enabled cryptographic hardware and configured Integrated Cryptographic Services Facility (ICSF) software. ICSF is the software interface to the cryptographic hardware. If you plan to run the Web server with cryptographic hardware capability, you can restrict the use of ICSF services. Do so by permitting user IDs and the WEBSRV ID to certain profiles in the CSFSERV general resource class. CSFSERV controls the use of ICSF software. If you have defined your Web server to execute with a nonzero user ID instead of WEBSRV, you can give the nonzero user ID READ access to CSFSERV. If you are using a security product other than RACF, refer to that product's documentation for instructions.

If you want to restrict the use of ICSF services, issue RACF commands similar to the ones in the examples below. If you have applications other than the Web server that are using ICSF, you must customize the examples. Otherwise the other applications will no longer have access to ICSF services.

- In this example the default access control user ID defined on the UserID directive is the surrogate ID of PUBLIC. The following example shows how to permit the WEBSRV ID and the PUBLIC ID access to profiles in CSFSERV.

```
SETROPTS RACLIST(CSFSERV) GENERIC(CSFSERV)
RDEFINE CSFSERV CSF* UACC(NONE)
PERMIT CSF%%C CLASS(CSFSERV) ID(WEBSRV PUBLIC) ACCESS(READ)
PERMIT CSFPK% CLASS(CSFSERV) ID(WEBSRV PUBLIC) ACCESS(READ)
PERMIT CSFCK% CLASS(CSFSERV) ID(WEBSRV PUBLIC) ACCESS(READ)
SETROPTS CLASSACT(CSFSERV)
SETROPTS RACLIST(CSFSERV) GENERIC(CSFSERV) REFRESH
```

- In this example the default access control user ID defined on the UserID directive is %%CLIENT%%. The following example shows how to give user IDs and the WEBSRV ID access to profiles in CSFSERV.

```
SETROPTS RACLIST(CSFSERV) GENERIC(CSFSERV)
RDEFINE CSFSERV CSF%%C UACC(READ)
RDEFINE CSFSERV CSFPK% UACC(READ)
RDEFINE CSFSERV CSFCK% UACC(READ)
SETROPTS CLASSACT(CSFSERV)
SETROPTS RACLIST(CSFSERV) GENERIC(CSFSERV) REFRESH
```

Related Information:

- For information on implementing hardware encryption and how to verify that the Web server is using hardware encryption, see “Hardware encryption” on page 65.

z/OS Workload Management

This section assumes you have already installed and set up Workload Management (WLM) on your z/OS system. If you plan to run the Web server in Scalable Server mode, use this procedure to enable the Web server to support WLM.

- Permit the Web server to WLM.

  The following example shows how to permit WEBSRV, which has a UID of 0, to WLM using RACF commands. If you have defined your Web server to execute with a nonzero user ID, you can instead permit the nonzero user ID to WLM. If you are using another security product, refer to that product’s documentation for instructions.

```
RDEFINE FACILITY MVSADMIN.WLM.POLICY UACC(NONE) NOTIFY(WEBSRV)
RDEFINE FACILITY BPX.WLMSERVER UACC(NONE) NOTIFY(WEBSRV)
PERMIT MVSADMIN.WLM.POLICY CLASS(FACILITY) ID(WEBSRV) ACCESS(UPDATE)
PERMIT BPX.WLMSERVER CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

- If you have defined the Web server with a nonzero user ID, you have the option to adjust the CPUTIMEMAX, ASSIZEMAX, and FILEPROCMAX settings in order to limit resource consumption. These initial settings were defined in “Defining the Web server with a nonzero user ID” on page 21.
  - **CPUTIMEMAX:** Start with the initial setting. The initial setting is the maximum value allowed. Adjust downward based on user experience.
  - **ASSIZEMAX:** Start with the initial setting. The initial setting is the maximum value allowed. Adjust downward based on user experience.
  - **FILEPROCMAX:** Start with five times the number of threads specified on the MaxActiveThreads directive in httpd.conf. Adjust upward or downward based on user experience.
- Define WLM resources

  For information on Web server configuration options, see “ApplEnv - Specify application environment for workload management” on page 555.

Related Information:

- For information on Web server modes of operation and WLM, see Chapter 13, “Managing your Web server” on page 225.

- For information on how to insure that your Queue Server procedure invokes your Queue Manager procedure correctly, see "Configuring your environment for workload management" on page 234.
- For more detailed information on WLM terminology and configuration, see *z/OS MVS Planning: Workload Management*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

System Management Facilities

The following example shows how to permit WEBSRV, which has a UID of 0, to SMF using RACF commands. If you have defined your Web server to execute with a nonzero user ID, you can instead permit the nonzero user ID to SMF. If you are using another security product, refer to that product's documentation for instructions.

```
RDEFINE FACILITY BPX.SMF UACC(NONE) NOTIFY(WEBSRV)
PERMIT BPX.SMF CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
```

Note: By default SMF capability is disabled in the Web server. If you wish to use SMF, you must enable it. For more information on SMF and how to enable it for the Web server, see "System Management Facilities" on page 277.

Lotus Notes access using RACF user IDs

The following example allows users to access the Web server from Lotus Notes using RACF user IDs. In this example, `userid` is the RACF user ID and `WEBSRV` is the user ID under which the Web server executes. `WEBSRV` is permitted to `userid` with READ access. If you have defined your Web server to execute with a nonzero user ID, you can instead permit the nonzero user ID READ access to `userid`.

1. Provide READ access to the RUSERMAP facility defined in RACF:

   ```
   PERMIT IRR.RUSERMAP CLASS(FACILITY)ID(WEBSRV)ACCESS(READ)
   ```

2. Assign Lotus Notes IDs to the RACF user profile using either ALTUSER or ADDUSER:

   ```
   ALTUSER userid LNOTES(SNAME('jsmith'))
   ADDUSER userid LNOTES(SNAME('jsmith'))
   ```

3. Allow the server surrogate use of the RACF IDs:

   ```
   RDEFINE SURROGAT BPX.SRV.userid UACC(NONE)
   PERMIT BPX.SRV.userid CLASS(SURROGAT)ID(WEBSRV)ACCESS(READ)
   SETROPTS RACLIST(SURROGAT) REFRESH
   ```

Related Information:

- Description of the HTTPD_local_security function in "Predefined functions and macros" on page 354

Step 4. Make TCP/IP configuration adjustments as needed

- Put TCP/IP data configuration in a location such as SYS1.TCPPARMS(TCPDATA) so z/OS UNIX sockets can find the name server.
- In your TCP/IP profile, reserve the port that the server uses to access z/OS UNIX (default is TCP 80 for a base server, TCP 443 for a secure server).

  ```
  ⋮
  PORT
  ⋮
  80 TCP OMVS          ; Base Server accesses OMVS
  ```

```
443 TCP OMVS         ; Secure Server accesses OMVS
⋮
```

- You may want to use TCP/IP to autostart the server PROC named IMWEBSRV from the TCP/IP profile. For autostart to be successful, you must have z/OS UNIX System Services started before you start TCP/IP.
- As part of TCP/IP, you can activate SNMP. Use the started task PROC, or activate SNMP from the console. For more information on SNMP, see "Simple Network Management Protocol" on page 253. For information on the data set search order, see Appendix G, "TCP/IP reference" on page 675.
- Use the TCP/IP onslookup command to verify that name resolution is working correctly, for example:
  1. From an OMVS command line, type:

     `onslookup` *`hostname`*
  2. Check that the returned IP address is correct:

     ```
     server :  your.server.name
     address : IP_address
     ```

For more information on the `onslookup` command, see the *eNetwork CS IP User's Guide*.

Step 5. Copy and customize the Web server PROC

Copy the IMWEBSRV sample provided in SYS1.SAMPLIB to a site procedure library, for example, SYS1.PROCLIB, and make the following changes as needed:

- If you changed the default names of the target libraries in IMWJALLO JOB supplied in SYS1.SAMPLIB, make the corresponding changes in the startup PROC.
- If you modify any RUNTIME options or set any server parameters, store the file containing these changes where you have your startup PROC.

To modify parameters and for an example of the IMWEBSRV PROC, see "IMWHTTPD program" on page 400.

The server needs to be assigned a user ID. This documentation assumes that the user ID of the server is WEBSRV. If you have defined your Web server to execute with a nonzero user ID, you can instead assign the nonzero user ID to the server.

For the IMWEBSRV cataloged procedure to obtain control with the desired user identity, you must add an entry to the RACF started procedures table, module ICHRIN03. This entry defines the user ID and group ID that the IMWEBSRV address space will be assigned.

To assign the user ID, WEBSRV, to the RACF started procedures table, module ICHRIN03, see the following example :

```
DC   CL8'IMWEBSRV' PROCEDURE NAME
DC   CL8'WEBSRV'   USERID (ANY RACF-DEFINED USER ID)
DC   CL8'IMWEB'    GROUP NAME OR BLANKS FOR USER'S DEFAULT GROUP
DC   XL1'00'       NOT TRUSTED
DC   XL7'00'       RESERVED
```

Note: Remember to increment the count of defined started procedures.

Or, you can add the PROC to the started task table. The following example provides a way to add the server PROC to the started task table using RACF commands:

```
RALTER STARTED IMWEBSRV.** STDATA(USER(WEBSRV))
SETROPTS RACLIST(STARTED) REFRESH
```

If you are defining the `IMWEBSRV.**` profile for the first time, use RDEFINE statements instead of RALTER statements.

If running with Workload Management, repeat the procedures described in this step for the Workload Management PROC, IMWIWM. See "IMWIWM PROC (workload management)" on page 409 for a sample workload management PROC.

If you are using a Queue Server procedure other than IMWIWM and a Queue Manager procedure other than IMWEBSRV, see "Configuring your environment for workload management" on page 234 to insure that your Queue Server procedure invokes your Queue Manager procedure correctly.

Step 6. Configure installed files by running setup.sh

You must run the supplied z/OS UNIX System Services shell script, setup.sh, to change the ownership of the installed files to your Web administration user ID, set up permissions, copy default configuration files, and configure languages.

Before running setup.sh

To run setup.sh, you must be a superuser or the SMP/E installer. This means you must have read and execute permission for setup.sh. The permission file for setup.sh should be 700. If someone other than the owner or superuser runs setup.sh, it may not execute correctly.

To execute setup.sh, you must be in the z/OS UNIX shell and utilities (by typing in OMVS in TSO ready). You cannot execute setup.sh in ISHELL. If you do not use the default Web administration user ID and group (WEBADM and IMWEB), then you must change these defaults coded in setup.sh to the specific names you used.

Tasks performed by setup.sh

Running setup.sh sets up your environment by performing the following tasks:

- Configures the Web server for languages by symbolically linking files to the parent directory.
- Backs up existing configuration files.

  If the Web server is installed in the default install path, **/usr/lpp/internet/**, your configuration files are copied to the **/etc/** directory. Otherwise, they are copied to the **/***install_path***/etc/** directory. Setupcfg.sh handles this for you.

  The following configuration and environment files are copied:
  - Web server configuration file, **httpd.conf**
  - Web server environment file, **httpd.envvars**
  - PICS configuration file, **ics_pics.conf**
  - Sample z/OS data set configuration file, **mvsds.conf**
  - FastCGI configuration file, **lgw_fcgi.conf**
  - Web Traffic Express configuration files, **javelin.conf** (proxy caching features configuration) and **socks.conf** (socksification configuration); if there is already a socks.conf file on your system, setup.sh creates a new file named socks.conf.exp.
- Checks to see if the IBM Software Development Kit (SDK) is installed on your system. If the SDK is not installed, setup.sh issues a message that no Java™ setup was done in the Web server httpd.envvars file.

Hints and tips

JAVA™_HOME: JAVA™_HOME points to the install root of the IBM Software Development Kit on your system. Some applications such as WebSphere Application Server and Java™ CGI's require that you add the install root to the paths of various environment variables in httpd.envvars. If you do not, the program in question will not be able to initialize.

If you set JAVA_HOME prior to running setup.sh, setup.sh will propagate the install root to the appropriate environment variables in httpd.envvars. If you do not set JAVA_HOME prior to running setup.sh, you can edit httpd.envvars and add the install root to the appropriate environment variables. For more information on the use of JAVA_HOME with WebSphere Application Server, refer to the Application Server documentation on the Web at URL:

```
http://www.ibm.com/software/webservers/appserv/library_390.html
```

Files are identical message when processing symbolic links: If you are migrating from a previous release of the Web server or if you run setup.sh more than once for a first-time install, you may see a message indicating that the source and target files are identical, for example:

```
FSUM8977  In: source "C/go98.jpg and target "./go98.jpg" are identical
```

You can ignore this message.

No such file or directory message when setting file properties: If you receive message EDC5129I for the httpd.conf.base, httpd.conf.secure, or servlet.conf file, you can ignore the message, for example:

```
chown: FSUM6188 stat file "samples/config/httpd.conf.base":
  EDC5129I No such file or directory
chown: FSUM6188 stat file "samples/config/httpd.conf.secure":
  EDC5129I No such file or directory
chown: FSUM6188 stat file "samples/config/servlet.conf":
  EDC5129I No such file or directory
```

Note: In this example each message appears on two lines for printing purposes. Each entry appears on one line in your output.

Running setup.sh

You may want to redirect setup.sh output messages to a file rather than the screen because they are easier to read. Otherwise, messages may scroll off the screen before you can read them. To redirect the error messages to a file, enter:

```
setup.sh 2> filename.filetype
```

If you use a shared Hierarchical File System (HFS), issue the **set** command on the z/OS UNIX services command line before issuing the **cd** command to change to your `sbin` subdirectory:

```
set -o logical
```

When you issue the **set** command, you insure that all your sample configuration files contain the path to your shared HFS once you run the `setup.sh` file.

Run the `setup.sh` file from the z/OS UNIX services command line by entering the following as two separate commands.

To set your configuration to English, enter:

```
cd /install_path/sbin
./setup.sh
```

To set your configuration to Japanese, enter:

```
cd /install_path/sbin
./setup.sh Ja_JP
```

Example of output messages:

```
*** (5697-D43) IBM HTTP Server 5.3 ***
Processing: default server install path is /usr/lpp/internet
             actual server install path is /usr/lpp/internet
       JDK install path is /usr/lpp/java/J1.3.1/IBM/J1.3

Processing: install English version of server
Processing: symbolic link of translatable files
Processing: symbolic link of files in ../server_root/Admin
Processing: symbolic link of files in ../server_root/admin-bin/webexec
Processing: symbolic link of files in ../server_root/Docs
Processing: symbolic link of files in ../server_root/pub
Processing: symbolic link of files in ../server_root/pub/reports
Processing: symbolic link of files in ../server_root/pub/reports/javelin
Processing: symbolic link of files in ../server_root/pub/PICSxmp
Processing: symbolic link of files in ../server_root/labels
Processing: symbolic link of files in ../server_root/Counters/Fonts
Processing: symbolic link of files in ../samples/config

Processing: set file properties
Processing: copying configuration and environment files
Processing: backup, copy and customize configuration file  (httpd.conf)
Processing: backup, copy and customize configuration file  (httpd.envvars)
Processing: backup, copy and customize configuration file  (ics_pics.conf)
Processing: backup, copy and customize configuration file (javelin.conf)
Processing: backup, copy and customize configuration file (lgw_fcgi.conf)
Processing: backup, copy and customize configuration file (mvsds.conf)
Processing: backup, copy and customize configuration file (socks.conf)
List of backup configuration files in: /etc

/etc/httpd.conf.0005121127
/etc/httpd.envvars.0005121127
/etc/ics_pics.conf.0005121127
/etc/javelin.conf.0005121127
/etc/lgw_fcgi.conf.0005121127
/etc/mvsds.conf.0005121127

JAVA_HOME is not pointing at the JDK; no Java setup
has been done in httpd.envvars.

Install Process Information:
-Language installed:  English
-Server install path is: /usr/lpp/internet
-JDK install path is: <JAVA_HOME>
-Server configuration files copied to: /etc and /usr/lpp/internet/etc
-Server configuration files default path is: /etc
*** setup.sh HAS COMPLETED ***
```

When running setup.sh from **/usr/lpp/internet/sbin**, you might receive the following error messages:

```
*** (5697-D43) IBM HTTP Server 5.3 ***
-setup.sh was not found in the current working directory.
-Execute setup.sh from the install path for this file.
-Change to the install path and execute setup.sh, again.
```

If you receive these messages, change the directory path to **/***install_path***/sbin** and execute setup.sh again.

Related Information:

- For detailed information on shared HFS, see *z/OS UNIX System Services Planning*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Step 7. Configure installed files on your target system by running setuptgt.sh (optional)

If you are running your server on a system that is different from the system that you installed the server on, you can run the supplied z/OS UNIX System Services shell script of setuptgt.sh on your target system in order to copy the default configuration files.

Before running setuptgt.sh

To run setuptgt.sh, you must be a superuser or the administrator (by default WEBADM). This means you must have read and execute permission for setuptgt.sh. The permissions for setuptgt.sh should be 700. If someone other than the owner or superuser runs setuptgt.sh, it may not execute correctly.

To execute setuptgt.sh, you must be in the z/OS UNIX shell and utilities (by typing in OMVS in TSO ready). You cannot execute setuptgt.sh in ISHELL. If you choose to use your Web administration user ID and group, but they are not the defaults of WEBADM and IMWEB, then you must change the defaults coded in setuptgt.sh to the specific names you used.

Tasks performed by setuptgt.sh

Running setuptgt.sh performs the following tasks:

- Backs up existing configuration files located in /etc.
- Copies configuration files from /usr/lpp/internet/samples/config to /etc and then customizes them. The configuration files are:
  - httpd.conf
  - httpd.envvars
  - ics_pics.conf
  - javelin.conf
  - lgw_fcgi.conf
  - mvsds.conf
  - socks.conf

    Note: If the socksification configuration file of socks.conf already exists, the Web server does not overlay the file. It creates a new file named socks.conf.exp before copying the sample socks.conf into it.

Running setuptgt.sh

You may want to re-direct setuptgt.sh output messages to a file rather than the screen because they are easier to read. Otherwise, messages may scroll off the screen before you can read them. To re-direct the error messages to a file, enter:

```
setuptgt.sh 2> filename.filetype
```

If you use a shared Hierarchical File System (HFS), issue the **set** command on the z/OS UNIX services command line before issuing the **cd** command to change to your `sbin` subdirectory:

```
set -o logical
```

Installing

When you issue the **set** command, you insure that all your sample configuration files contain the path to your shared HFS once you run the `setuptgt.sh` file.

Run the `setuptgt.sh` file from the z/OS UNIX command line by entering the following as two separate commands:

```
cd <mountpoint>/usr/lpp/internet/sbin
./setuptgt.sh
```

```
*** (5697-D43) IBM HTTP Server 5.3 for OS/390 ***
Processing: server install path is /usr/lpp/internet
            JDK install path is /usr/lpp/java

Processing: copying configuration and environment files

Processing: backup, copy and customize configuration file
(httpd.conf)

Processing: backup, copy and customize configuration file
(httpd.envvars)

Processing: backup, copy and customize configuration file
(ics_pics.conf)

Processing: backup, copy and customize configuration file
(javelin.conf)

Processing: backup, copy and customize configuration file
(lgw_fcgi.conf)

Processing: backup, copy and customize configuration file
(mvsds.conf)

Processing: adding Java paths to httpd.envvars

Processing: backup, copy and customize configuration file
(socks.conf)

List of backup configuration files in: /etc

/etc/httpd.conf.0009211125     /etc/javelin.conf.0009211125
/etc/httpd.envvars.0009211125  /etc/lgw_fcgi.conf.0009211125
/etc/ics_pics.conf.0009211125  /etc/mvsds.conf.0009211125

Install Process Information:

- Server install path is: /usr/lpp/internet
- JDK install path is: /usr/lpp/java
```

When running setuptgt.sh from **/usr/lpp/internet/sbin**, you might receive the following error messages:

```
*** (5697-D43) IBM HTTP Server 5.3 for OS/390 ***
-setuptgt.sh was not found in the current working directory.
-Execute setuptgt.sh from the install path for this file.
-Change to the install path and execute setuptgt.sh, again.
```

If you receive these messages, change the directory path to **/***actual_install_path***/sbin** and execute setuptgt.sh again.

Related Information:

- For detailed information on shared HFS, see *z/OS UNIX System Services Planning*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Step 8. Customize your Web server configuration file httpd.conf

The sample Web server configuration allows you to access pages in the following directories:

- */install_path*/**server_root/pub** (default Front Page, Frntpage.html and other Web server pages)
- */install_path*/**server_root/Admin** (Configuration and Administration Forms)

You can customize your Web server configuration by editing the httpd.conf file directly or by using the Configuration and Administration Forms graphical user interface. For more information about these options, see "How do I configure the Web server?" on page 50.

Using a Web server user ID other than WEBADM

After installation, your Web server has one authorized user ID that can be used to access the Configuration and Administration Forms. By default, the authorized user ID is WEBADM.

If you selected a different user ID during Web server installation:

1. Edit the httpd.conf file and change all occurrences of WEBADM and webadm to the user ID you selected.
2. Change the owner of files in the following directory to the user ID you specified:

   ```
   /install_path/server_root/admin-bin
   ```

The *install_path* is the root directory of your Web server installation. This value is either the default install path (**/usr/lpp/internet**) or the install path you specified during installation.

Changing default directories for logs and reports

The sample configuration file assumes you want to use the following directories for logging and reporting:

```
/install_path/server_root/logs
/install_path /server_root/pub/reports
```

Create these directories, or change the configuration file to point to an existing directory, before starting the IMWEBSRV PROC. Consider creating a new HFS to mount for logging and reporting. (See Chapter 11, "Customizing logs and reports" on page 189 for information about logging and reporting.)

Step 9. Customize your Web server environment variables file

Spawn programs into separate address spaces from that of the Web server. Specify **_BPX_SHAREAS=NO** in the `httpd.envvars` file to prevent z/OS UNIX System Services from loading spawned programs into the Web server address space. z/OS UNIX System Services can load spawned programs into the Web server address space if you do one of the following:

- Specify **_BPX_SHAREAS=YES** in the `httpd.envvars` file.
- Leave the **_BPX_SHAREAS** environment variable out of the `httpd.envvars` file, but specify **_BPX_SHAREAS=YES** elsewhere.

Examples of spawned programs are CGI's, and the Web server's logging and reporting programs. Since these programs are not marked as program controlled, the Web server receives error code 02AF, (address space dirty), if these programs load into the Web server address space.

Note: APAR PQ38585 ships a new sample `httpd.envvars` file for HTTP Server version 5.1 and later. This sample specifies **_BPX_SHAREAS=NO**.

Step 10. Set up security

For configuration information and examples, see Chapter 8, “Setting up a secure server” on page 53.

Step 11. Set up proxy server support (optional)

To set up a proxy server, see Chapter 16, “Running your server as a proxy” on page 303.

Step 12. Install and configure WebSphere Application Server (optional)

For planning, installation, and configuration information, refer to the Application Server documentation. You can access the latest Application Server documentation on the Web at URL:

`http://www.ibm.com/software/webservers/appserv/library_390.html`

What’s next?

Now that your Web server is installed and customized, you can start the server, view the server’s default Front Page, and change your server’s default configuration. To get started, go to Chapter 4, “Starting the server” on page 39.

Part 3. Using

Chapter 4. Starting the server

Performance considerations when starting the Web server 39
Starting the server from the start-up PROC 39
Starting the server using the httpd command . . . 39
Starting the server from the z/OS UNIX shell . . . 39
Starting multiple instances of the server 40
Starting the server with Workload Management running. 40
Letting WLM start the queue server address space 40
Prestarting the queue server address space . . . 40
Restarting the server 41

Performance considerations when starting the Web server

It is recommended that production Web servers be started using the method described in "Starting the server from the start-up PROC". It may be difficult to achieve desired performance results if you are running the Web server from the z/OS UNIX shell.

If you choose to start the Web server from a UNIX shell, review the information and recommendations in the Tuning section of the WebSphere Troubleshooter for z/OS at URL:

`http://www.ibm.com/software/webservers/httpservers/doc/v51/ttunc.htm`

Starting the server from the start-up PROC

You can start the server using the start-up PROC, IMWEBSRV. Make sure it is a member in a PROCLIB data set. Enter the following command from the z/OS operator console:

```
s IMWEBSRV
```

For a sample IMWEBSRV PROC and variables used in the PROC, see "IMWHTTPD program" on page 400.

Starting the server using the httpd command

You can start the server using the httpd command. For more information, see "httpd command" on page 393.

Starting the server from the z/OS UNIX shell

Note: Before using this method to start your Web server, see "Performance considerations when starting the Web server".

To start the server, you must be in `/usr/lpp/internet/sbin` or provide the fully-qualified path on the httpd command (The following example assumes you are in `/usr/lpp/internet/sbin`.)

You can start the server from the z/OS UNIX shell environment using the default server configuration file /etc/httpd.conf by entering:

```
httpd
```

This starts the server with the current configuration settings.

Starting multiple instances of the server

You can start multiple instances of the server, but each instance must listen on a separate port or IP address. All instances can be started by entering the following command from a command prompt:

```
httpd -r other_rule_file
```

For *other_rule_file*, enter the configuration file that specifies the individual port.

To start the secure server on the same machine but on a different port, enter the following command from a command prompt:

```
httpd -p current_port -sslport new_port
```

For *current_port*, enter the port number the server is using. For *new_port*, enter the port number of the new port for the server.

If Workload Management is enabled, unique subsystem names (-SN) must be defined. It is also recommended that you run under a unique server root and a unique access report root.

You can also use a start-up PROC or batch JCL to start multiple instances of the server by making similar changes.

Starting the server with Workload Management running

When Workload Management is started, temporary files are created and stored in /tmp. These files do not get deleted if the cancel command is used to stop the server. Therefore, use the stop command for proper clean-up.

To minimize any initial start-up costs (when running WLM), you can prestart your server address space once you receive the message indicating the server is ready.

Letting WLM start the queue server address space

- Under the Application Environment section of the WLM panels, you have specified the PROC which is to be started when there is work on that queue. A sample workload management PROC, IMWIWM, is shipped with the product. See “Workload Management Enablement for the Web server” on page 228 for more information about Workload Management panels.
- Specify the start parameters on the Workload Management ApplEnv panel:

  ```
  IWMSN=&IWMSSNM, IWMAE=ApplEnv_name
  ```

- Start the QueueManager address space like you usually do, using JCL or a console command. When work is put on a queue, WLM will automatically start the corresponding PROC and start the address space for the queue server.

Prestarting the queue server address space

You can prestart the server by doing one of the following:

- The Queue Manager address space may be started by submitting the following job, or using the line command under the z/OS UNIX shell: (-SN should be specified in the PARM statement to indicate the subsystem name used.)

  ```
  //RUNHTTP jobcard
  //STEP1 IMWEBSRV,
  //      ICSPARM='-SN WEBSN1'
  ```

- Enter the following console command:

```
S IMWEBSRV,ICSPARM='-SN WEBSN1'
```

After the Queue Manager address space has been started, you need to start the Queue Server address space that correlates with the subsystem instance.

You can do this by doing one of the following:

- The QueueServer address space must be started by submitting the following job: (-SN and -AE should be specified in the PARM statement. The subsystem name specified should correlate with the one specified by the Queue Manager.)

  ```
  //RUNSRVR jobcard
  //STEP1 IMWEBSRV,
  //      ICSPARM='-SN WEBSN1 -AE WEBHTML'
  ```

- Enter the following console command:

  ```
  S IMWEBSRV,ICSPARM='-SN WEBSN1 -AE WEBHTML'
  ```

Note that WEBHTML is the name of the appropriate queue to be started.

Restarting the server

You can restart the server in the following ways:

- If you are using Workload Management and need to restart the server with a new configuration file, the workload manager marks the server space and completes work that is currently running. Workload Management starts a new server, finds the new configuration file, and identifies a new set of server spaces.
- Restart the server using the z/OS operator console MODIFY command, F IMWEBSRV,APPL=-restart
- Send SIGHUP to the server process ID from the z/OS UNIX services shell environment.

  Use the z/OS UNIX services shell command **kill** or the server **wwwcmd** command to send SIGHUP to the server process. Refer to the *z/OS UNIX System Services Command Reference* for more information about the **kill** command. For information on the **wwwcmd** command, see "wwwcmd command" on page 409.
- Use the server command **httpd** with the -restart option. For more information, see "httpd command" on page 393.
- Use the MVS cataloged procedure, IMWEBSRV, with the -restart option.
- Use the Restart Server button on the Remote Configuration and Administration forms.

Chapter 5. Viewing the Web server's default Front Page

After you install and start the Web server, you can use any compatible Web browser to view the server's default Front Page at URL:

```
http://your.server.name/Frntpage.html
```

For *your.server.name*, enter your server's fully qualified host name (for example, **myhost.raleigh.ibm.com**).

Note: For information on browser requirements and tips, see "Compatible and tested browser levels" on page 3.

The default Front Page provides the following links:

- **CONFIGURATION AND ADMINISTRATION FORMS** to set up, configure, and administer the IBM HTTP Server
- **IBM HTTP SERVER WEB SITE** to find the most current information
- **HOW DO I GET STARTED?** to get help and examples for basic configuration tasks
- **INFORMATION TO HELP YOU** to access the online documentation

From the default Front Page of the Web server, you can link to online versions of the following information:

- *HTTP Server Planning, Installing, and Using*
- Application Server documentation
- *WebSphere Troubleshooter for z/OS and OS/390*
- *IBM Web Traffic Express for Multiplatforms User's Guide V1.0*

The most current information is available on the HTTP Server Web site at URL:

You can access the latest Application Server documentation on the Web at URL:

```
http://www.ibm.com/software/webservers/appserv/library_390.html
```

Chapter 6. Stopping the server

You can stop the server if you know the WEBSRV user ID or have root authority. The server can be stopped the same way you stop any other z/OS UNIX process:

- Use the z/OS operator console STOP command, **p IMWEBSRV**.
- Use the z/OS operator console CANCEL command, **c IMWEBSRV**.
- Send a process termination signal to the server process ID from the z/OS UNIX shell environment. Both the SIGTERM signal and the SIGKILL signal can be sent, but the effect differs slightly.

  SIGTERM causes the server to perform a shutdown which allows outstanding client requests more time to complete before stopping. SIGKILL causes the server to terminate immediately.

  Use the z/OS UNIX MVS shell command **kill** or the server command **wwwcmd** to send SIGTERM or SIGKILL to the server process.

  Refer to the *z/OS UNIX System Services Command Reference* for more information about the **kill** command. For more information about the **wwwcmd** command, see "wwwcmd command" on page 409.

Note: The cancel command does not perform clean up of server files. Extra files may remain in the /tmp directory.

Stopping the server

Part 4. Basic Configuration

Chapter 7. Getting started

How do I start serving pages? 49
How do I serve directory listings? 49
How do I configure the Web server? 50
Editing the configuration file 50
Using the Configuration and Administration forms 50
How do I control access to server administration functions? 51
Controlling access to the Configuration and Administration forms 51
Copying files 51
What files should I back up? 52

After you have the server installed and running, you can start serving HTML pages and Web sites. As you become more familiar with the server, you will probably want to modify it by changing its configuration to meet your own particular needs.

This chapter helps you get started serving pages. It also shows you how to change your server's configuration using the Configuration and Administration forms or by editing the configuration file, and how to control access to your server's administration functions.

How do I start serving pages?

The HTTP Server can serve files of many file types. The most common are HTML files, also called HTML pages or documents, that contain text. They often imbed GIF or JPEG files to include graphic images.

If you point your browser to the host where you installed your server and you can get to the Front Page, your server is ready to go. Try going to **http://***your.server.name***/** and make sure this works. Now you can add your own HTML documents, including welcome or home pages, and serve them.

Detailed instructions and information for this task are available in the online help available from the default Front Page of your server or from the Configuration and Administration Forms user interface. To access the online help, use either of the following methods:

- From the default Front Page of your server, click **HOW DO I GET STARTED?**, then click **How do I?** to display the list of tasks.
- From the Configuration and Administration Forms, click the question mark in the upper right hand corner, then click **How do I?** to display the list of tasks.

How do I serve directory listings?

When the server receives a directory request, it can serve either a welcome page from that directory or a formatted list of links to the files in the directory. The server's default setting always returns a welcome page. You can use the Configuration and Administration forms to change this and serve directory listings for all directory requests or only for selected ones. You can also modify the contents of the directory listings you serve.

Detailed instructions and information for this task are available in the online help available from the default Front Page of your server or from the Configuration and Administration Forms user interface. To access the online help, use either of the following methods:

- From the default Front Page of your server, click **HOW DO I GET STARTED?**, then click **How do I?** to display the list of tasks.
- From the Configuration and Administration Forms, click the question mark in the upper right hand corner, then click **How do I?** to display the list of tasks.

How do I configure the Web server?

You can configure your Web server by editing the configuration file directly or use the Configuration and Administration Forms graphical user interface. After installation, your Web server has one authorized user name that can be used to access the Configuration and Administration Forms. By default, the authorized user name is WEBADM.

Editing the configuration file

By default, the configuration file is named /etc/httpd.conf. See "httpd command" on page 393 for information on using a different configuration file when starting the server.

The configuration file contains statements called ***directives***. You change your configuration by editing the configuration file, updating the directives, and saving your changes. Appendix B, "Configuration directives" on page 417 describes each of the configuration file directives.

Using the Configuration and Administration forms

The server comes with Configuration and Administration forms. These forms are a combination of CGI programs and HTML forms that provide an easy way for you to configure your server or to view your server's current configuration settings.

Once the server is running, you can use the Configuration and Administration forms from any Web browser to configure your server.

To use the Configuration and Administration forms:

1. Disable caching on your browser. Also, if you are configuring your server remotely from a browser that uses that specific server as its proxy server, you should disable the proxy server setting on your browser.
2. Using your browser, go to the server's Front Page by typing the following URL:

   http://*your.server.name***/**

   where *your.server.name* is the fully qualified name of your host. For example, http://www.ibm.com/
3. Click **Configuration and Administration Forms**.
4. If you have not used the Configuration and Administration forms since starting your browser, you will be prompted for a user name and password.

   After you enter an authorized user name and password, you go to the Configuration and Administration forms page.
5. From the Configuration and Administration forms page, you can link to each of the input forms by clicking on the form name.

   When you go to a form, it is displayed with the current configuration values in its input fields. (If you haven't changed your configuration since installation, these are the values that have been set in the shipped configuration file.)
6. From any form, enter information about how you want to configure that particular part of your server.

Each form provides instructions to assist you in deciding what changes to make. For further information, click the question mark (?) in the upper right-hand corner of the page and choose from three kinds of help:

- **Field descriptions** for information about each field on the form
- **How do I?** for information about configuration tasks
- **Index** for an index of all the help topics

7. After you fill in the form, click **Submit** to update the server configuration with the changes you made. The **Submit** button is located below the input fields on each form.

   If you decide you do not want to use the changes you made to the form, click **Reset**. This returns the fields on the form to the values they had when you first came to the form.
8. If you clicked **Submit**, the server re-displays the form and provides a status message at the top of the page. This message tells you if your requested configuration changes completed successfully or, if not, which errors occurred. While the form is displayed, you can fix the errors and resubmit it.
9. After a successful completion, click the restart button in the upper right hand corner of the page to restart your server. When it is restarted, the server completes in-process requests, stops accepting requests, and reloads the changed configuration file. After the changed configuration file has been reloaded, the server accepts requests again.

   Note: After submitting changes on the **Basic** configuration form, you must stop your server and start it again in order for your changes to take effect. In this case, restarting the server from the form is not effective.

   For instructions on stopping and starting your server, see Chapter 4, "Starting the server" on page 39.

How do I control access to server administration functions?

Controlling access to the Configuration and Administration forms

During installation, you created a user ID (recommended name `WEBADM`) and group ID (recommended name `IMWEB`) to configure and administer the server. This user ID and password are authorized to access the Configuration forms.

If you intend to run the HTTP Server as a secure server (for example, with SSLmode on), certificates and certificates requests are electronically mailed to a certificate authority (CA) from this user ID.

Copying files

Be careful when you copy files to ensure you give the correct permissions to the `WEBADM` user ID and the `IMWEB` group ID. For example, `WEBADM` needs write permission to modify ACL files and, for a secure server, write permission to modify key database files (*.kdb).

If the user ID or group ID changes, you need to switch any ACL files to the new user ID or group ID. For a secure server, in addition to switching ACL files, you also need to switch key database files (*.kdb) and database password files (*.sth). If you are using %%SAF%% for security, you do not need to switch ACL files.

What files should I back up?

It is recommended that you periodically back up the following files:

- All configuration files
- Environment variables file (httpd.envvars)
- Password files
- Key database files
- Signed certificates and, optionally, certificate requests
- Group files
- ACL files
- Content files

Chapter 8. Setting up a secure server

Security concepts 55
What is a secure connection? 55
What is encryption? 56
What is authentication? 57
What is SSL? 58
What is a Public Key Infrastructure? 58
What is LDAP? 59
X.500 overview 60
LDAP overview 60
Security options for the HTTP Server 61
Options for setting up secure connections . . . 61
SSL support for multiple IP addresses 62
Support for specifying the encryption level to be used 62
Certificate authorities supported by the HTTP Server 62
Buying a certificate from an external commercial CA 62
Acting as your own CA 63
Encryption support for the HTTP Server. 64
Supported public-private key sizes 64
North American edition (U.S. and Canada) . . 64
International export edition 64
Supported SSL cipher specifications 64
North American edition (U.S. and Canada) . . 65
International export edition 65
Hardware encryption 65
How hardware encryption affects Web server performance 65
Basic information needed to implement hardware encryption 66
How to check whether hardware encryption is used for Web server encryption. 67
Checklist for setting up a secure server 68
Examples: Setting up secure connections. 69
Important notes 69
Which example should I use? 70
Terms used in these examples 73
Example 1A for gskkyman: Setting up secure connections using a self-signed server certificate (z/OS Version 1 Release 4 or later) 75
Summary of tasks 75
Step 1. Create the server key database . . . 75
Step 2. Create your self-signed server certificate 76
Step 3. Register the server key database with the Web server 77
Step 4. Use server security defaults 77
Step 5. Verify that you can establish a secure connection with the server 78
Example 2A for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 4 or later) 78
Summary of tasks 78
Step 1. Set up the server key database . . . 79
Step 2. Send your server certificate request to an external commercial certificate authority. . 80
Step 3. If necessary, add the CA to the list of trusted CAs on the server 81
Step 4. Receive your signed server certificate into the server key database 82
Step 5. Register the server key database with the server 83
Step 6. Use server security defaults 83
Step 7. Verify that you can establish a secure connection with the server 83
Step 8. Optionally, set up client authentication using client certificates 84
Example 3A for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 4 or later) . 84
Summary of tasks 84
Step 1. Set up the CA key database 85
Step 2. Set up the server key database . . . 86
Step 3. Sign the server certificate 88
Step 4. Receive the signed server certificate into the server key database 89
Step 5. Add your CA name to the list of trusted CAs for your browser 89
Step 6. Register the server key database with the Web server 90
Step 7. Use server security defaults 90
Step 8. Verify that you can establish a secure connection with the server 90
Step 9. Optionally, set up client authentication 90
Example 4A for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 4 or later) 90
Summary of tasks 91
Step 1. Update server configuration directives 91
Step 2. Insure the CA certificate is in the client browser 93
Step 3. If acting as your own CA, provide options for obtaining client certificates . . . 93
Step 4. Map a client certificate to Resource Access Control Facility 96
Step 5. Verify that the client can access a protected page with the client certificate. . . 98
Example 1 for gskkyman: Setting up secure connections using a self-signed server certificate (z/OS Version 1 Release 3 or earlier) 98
Summary of tasks 99
Step 1. Create the server key database . . . 99
Step 2. Create your self-signed server certificate. 100
Step 3. Register the server key database with the Web server 100
Step 4. Use server security defaults 101
Step 5. Verify that you can establish a secure connection with the server 101

Security

Example 2 for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 3 or earlier) . . . 101
Summary of tasks . . . 102
Step 1. Set up the server key database . . . 102
Step 2. Send your server certificate request to an external commercial certificate authority . 103
Step 3. If necessary, add the CA to the list of trusted CAs on the server . . . 104
Step 4. Receive your signed server certificate into the server key database . . . 105
Step 5. Register the server key database with the server . . . 106
Step 6. Use server security defaults . . . 106
Step 7. Verify that you can establish a secure connection with the server . . . 106
Step 8. Optionally, set up client authentication using client certificates . . . 107
Example 3 for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 3 or earlier) . . . 107
Summary of tasks . . . 107
Step 1. Set up the CA key database . . . 108
Step 2. Set up the server key database . . . 109
Step 3. Sign the server certificate . . . 111
Step 4. Receive the signed server certificate into the server key database . . . 111
Step 5. Add your CA name to the list of trusted CAs for your browser . . . 112
Step 6. Register the server key database with the Web server . . . 112
Step 7. Use server security defaults . . . 112
Step 8. Verify that you can establish a secure connection with the server . . . 113
Step 9. Optionally, set up client authentication . . . 113
Example 4 for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 3 or earlier) . . . 113
Summary of tasks . . . 114
Step 1. Update server configuration directives 114
Step 2. Insure the CA certificate is in the client's browser. . . . 115
Step 3. If acting as your own CA, provide option(s) for obtaining client certificates . . 116
Step 4. Map a client certificate to RACF . . 119
Step 5. Verify that the client can access a protected page with the client certificate . . 120
Example 5 for RACDCERT: Setting up secure connections using a self-signed server certificate. 120
Summary of tasks . . . 121
Step 1. Create the server key ring. . . . 121
Step 2. Create your self-signed server certificate. . . . 122
Step 3: Connect your signed server certificate to the key ring . . . 122
Step 4: Permit the Web server ID to access the key ring through DIGTCERT . . . 123
Step 5. Register the server key ring with the Web server . . . 123
Step 6. Use server security defaults . . . 123
Step 7. Verify that you can establish a secure connection with the server . . . 123
Example 6 for RACDCERT: Setting up secure connections using certificates signed by an external commercial CA . . . 124
Summary of tasks . . . 124
Step 1. Create the server key ring. . . . 125
Step 2. Create your server certificate request 125
Step 3. Send your server certificate request to an external commercial CA . . . 126
Step 4. Insure that your CA certificate is in RACF and marked with a status of TRUST . 126
Step 5. Add your server certificate to the key ring . . . 127
Step 6: Connect your signed server certificate to the server key ring. . . . 128
Step 7. Permit the Web server ID to access the key ring through DIGTCERT . . . 129
Step 8. Register the server key ring with the server . . . 129
Step 9. Use server security defaults . . . 129
Step 10. Verify that you can establish a secure connection with the server . . . 130
Step 11. Optionally, set up client authentication using client certificates . . . 130
Example 7 for RACDCERT: Setting up secure connections using certificates signed by an internal CA . . . 130
Summary of tasks . . . 131
Step1. Create a self-signed CA certificate . . 131
Step 2. Set up the server key ring. . . . 132
Step 3. Create and sign your server certificate 132
Step 4. Connect your signed server certificate to the server key ring. . . . 133
Step 5. Add your CA name to the list of trusted CAs for your browser . . . 133
Step 6. Permit the Web server ID to access the key ring through DIGTCERT . . . 134
Step 7. Register the server key ring with the Web server . . . 135
Step 8. Use server security defaults . . . 135
Step 9. Verify that you can establish a secure connection with the server . . . 135
Step 10. Optionally, set up client authentication . . . 135
Example 8 for RACDCERT: Setting up client authentication using client certificates signed by an internal CA . . . 135
Summary of tasks . . . 137
Step 1. Update server configuration directives 137
Step 2. Insure the CA certificate is in the client's browser. . . . 138
Step 3. Create the client certificate and associate it with a RACF user ID . . . 138
Step 4. Add the signed client certificate to the client's browser. . . . 139
Step 5. Verify that the client can access a protected page with the client certificate . . 140

Example 9 for RACDCERT: Setting up client authentication using client certificates signed by an external CA 140
Summary of tasks 141
Step 1. Update server configuration directives 142
Step 2. Insure the CA certificate is in the client's browser. 143
Step 3. Obtain client certificate 143
Step 4. Map a client certificate to a RACF user ID 143
Step 5. Verify that the client can access a protected page with the client certificate . . 145
Example 10A: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1 Release 4 or later) 145
Summary of tasks 146
Step 1. Create the server key ring in RACF 146
Step 2. Insure that your CA certificate exists in RACF and has a status of TRUST. . . . 147
Step 3. Insure that your server certificate and private key exist in RACF 149
Step 4. Permit the ID used to start the server to access the key ring. 150
Step 5. Register the server key ring with the Web server 151
Step 6. Use server security defaults 151
Step 7. Verify that you can establish a secure connection with the server 151
Step 8. Set up client authentication as an option by using client certificates. 151
Example 10: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1 Release 3 or earlier) 151
Summary of tasks 152
Step 1. Create the server key ring in RACF 153
Step 2. Insure that your CA certificate exists in RACF and has a status of TRUST. . . . 153
Step 3. Insure that your server certificate and private key exist in RACF 155
Step 4. Permit the ID used to start the server to access the key ring. 156
Step 5. Register the server key ring with the Web server 157
Step 6. Use server security defaults 157
Step 7. Verify that you can establish a secure connection with the server 157
Step 8. Set up client authentication as an option by using client certificates. 157
Related information 157
Certificate file types generated by the gskkyman utility 157
Key database password 158
asn.1 encoding and decoding error 158
Related documentation and URLs 159
z/OS documentation 159
Security documentation 159
Setting up SSL support for multiple IP addresses 159
Changing the default order of encryption levels used by the Web server 160

Security concepts

This section provides an overview of security concepts. If you are already familiar with basic security concepts and want to get started, go to "Examples: Setting up secure connections" on page 69.

What is a secure connection?

The rapid growth of e-business on the Internet has led to an increasing demand for secure network communications. In addition, communications over private networks often contain confidential information that needs to be protected.

A secure connection has the following characteristics:

Access control
Resources can be protected and accessed only by authorized parties. Restricting access on the basis of passwords, IP address, host names, or SSL client authentication ensures access control.

Authenticity
You know who you are talking to and that you can trust that person. Authentication, using digital signature and digital certificates, ensures authenticity.Messages are not altered while being transmitted. Without information integrity, you have no guarantee that the message you sent matches the message received. Digital signature ensures integrity.

Information integrity
Messages are not altered while being transmitted. Without information

integrity, you have no guarantee that the message you sent matches the message received. Digital signature ensures integrity.

Privacy and confidentiality
Information conveyed from party to party during a transaction remains private and cannot be read even if it gets into the wrong hands. Encryption ensures privacy and confidentiality.

What is encryption?

Encryption in its simplest form is scrambling a message so that it cannot be read until it is unscrambled later by the receiver. The sender uses an algorithmic pattern (or key) to scramble (or encrypt) the message. The receiver has the decryption key. Encryption ensures privacy and confidentiality in transmissions sent over the Internet.

There are two kinds of keys that can be used for encryption:

Asymmetric keys

With asymmetric keys, you create a key pair. The key pair is made up of a public key and a private key, which are different from each other. The private key holds more of the secret encryption pattern than the public key. Your private key should not be shared with anyone.

The server uses its private key to sign messages to clients. The server sends its public key to clients so that they can encrypt messages to the server, which the server decrypts with its private key. Only you can decrypt a message that has been encrypted with your public key, because only you have the private key. Key pairs are stored in a key database which is protected by a password.

Symmetric keys

Symmetric keys follow an age-old model of the sender and receiver sharing some kind of pattern. This same pattern is then used by the sender to encrypt the message and by the receiver to decrypt the message.

The risk involved with symmetric keys is that you have to find a safe transportation method to use when sharing your secret key with the people you want to communicate with.

The Secure Sockets Layer (SSL) protocol uses both asymmetric and symmetric key exchange. Asymmetric keys are used for the SSL handshake. During the handshake the master key, encrypted with the receiver's public key, is passed from the client to the server. The client and server make their own session keys using the master key. The session keys are used to encrypt and decrypt data for the remainder of the session. Symmetric key exchange is used during the exchange of the cipher specification (or encryption level) used.

To send its public key to clients, the server needs a digital certificate. This certificate is issued by a trusted third-party called a certificate authority (CA) who verifies the identity of the server.

Related Information:

- For more information about the SSL protocol, see "What is SSL?" on page 58.
- For details on supported key sizes, see "Encryption support for the HTTP Server" on page 64.

What is authentication?

Authentication is the process used to verify identity, so that you can ensure that others are who they say they are. There are two ways that the server uses authentication:

Digital signature

A digital signature is a unique mathematically computed signature that ensures accountability. A digital signature is similar to a credit card with your picture on it. To verify the identity of the person sending you a message, you look at the sender's digital certificate.

Digital certificate

A digital certificate (or digital ID), is like a credit card with a picture of the bank president with his arm around you. A merchant trusts you more because not only do you look like the picture on the credit card, the bank president trusts you, too.

You base your trust for the authenticity of the sender on whether you trust the third party (a person or agency) that certified the sender. The third party who issues digital certificates is called a certificate authority (CA) or certificate signer.

A digital certificate contains:
- The public key of the person being certified
- The name and address of the person or organization being certified, also known as the Distinguished Name
- The digital signature of the CA
- The issue date
- The expiration date

You enter your Distinguished Name as part of requesting a certificate. The digitally-signed certificate includes not only your own Distinguished Name but the Distinguished Name of the CA.

You can request one of the following certificates:
- A server certificate to do commercial business on the Internet from VeriSign or some other CA. For a list of supported CAs, see "Buying a certificate from an external commercial CA" on page 62.
- A server certificate that you create for your own private Web network. For more information, see "Acting as your own CA" on page 63.

CAs broadcast their public key and Distinguished Name bundled together so that people will add them to their Web servers and browsers as a trusted CA certificate. When you designate the public key and certificate from a CA to be a trusted CA certificate, this means that your server trusts anyone who has a certificate from that CA. You can have many trusted CAs as part of your server. The HTTP Server includes several default trusted CA certificates. You can add or remove trusted CAs as needed using the z/OS System SSL gskkyman utility or RACF.

In order to communicate securely, the receiver in a transmission must trust the CA who issued the sender's certificate. This is true whether the receiver is a Web server or browser. When a sender signs a message, the receiver must have the corresponding CA-signed certificate and public key designated as a trusted CA certificate.

Related Information:

- For more information on CAs and a list of default trusted CAs, see "Certificate authorities supported by the HTTP Server" on page 62.

What is SSL?

Secure Sockets Layer (SSL) is a communications protocol that provides secure communications over an open communications network, such as the Internet. z/OS System SSL provides SSL support for the HTTP Server.

The SSL protocol was developed by Netscape Communications Corporation. SSL provides data privacy and integrity as well as server and client authentication.

Once your server has a digital certificate, SSL-enabled browsers like Netscape Navigator and Microsoft Internet Explorer can communicate securely with your server using SSL. With SSL, you can easily establish a security-enabled Web site on the Internet or on your private intranet. A browser that does not support HTTP over SSL will not be able to request URLs using HTTPS. The non-SSL browsers will not allow submission of forms that need to be submitted securely.

SSL uses a security handshake to initiate a secure connection between the client and the server. During the handshake, the client and server agree on the security keys they will use for the session and the algorithms they will use for encryption. The client authenticates the server; optionally, the server can request the client's certificate. After the handshake, SSL is used to encrypt and decrypt all of the information in both the **https** request and the server response, including:
- The URL the client is requesting
- The contents of any form being submitted
- Access authorization information like user names and passwords
- All data sent between the client and the server

HTTPS is a unique protocol that combines SSL and HTTP. You need to specify **https://** as an anchor in HTML documents that link to SSL-protected documents. A client user can also open a URL by specifying **https://** to request an SSL-protected documents.

Because HTTPS (HTTP + SSL) and HTTP are different protocols and use different ports (443 and 80, respectively), you can run both SSL and non-SSL requests at the same time. As a result, you can choose to provide information to all users using no security, and specific information only to browsers who make secure requests. This is how a retail company on the Internet can allow users to look through the merchandise without security, but then fill out order forms and send their credit card numbers using security.

Related Information:
- For more information on the SSL Version 3 specification, go to URL http://home.netscape.com/eng/ssl3/.
- For more information on SSL, go to the IBM Security Library Web site at URL http://www.ibm.com/security/library/.

What is a Public Key Infrastructure?

A Public Key Infrastructure (PKI) is a system of digital certificates, certificate authorities, registration authorities, certificate management service, and X.500 directories that verify the identity and authority of each party involved in any transaction over the Internet. These transactions may be financial or may involve

any operation where identity verification is required, such as confirming the origin of proposal bids or author of e-mail messages.

A PKI supports the use of certificate revocation lists (CRLs). A certificate revocation list is a list of certificates that have been revoked. CRLs provide a more global method for authenticating a client's identity by certificate, and can also be used to verify the validity of trusted CA certificates.

CRLs and trusted CA certificates are stored and retrieved from an X.500 directory server. The protocols used for storing and retrieving information from an X.500 directory server are Directory Access Protocol (DAP) and Lightweight Directory Access Protocol (LDAP). The HTTP Server supports LDAP.

Information can be distributed on multiple directory servers over the Internet and intranets, thus allowing an organization to manage certificates, trust policy, and CRLs from either a central location or in a distributed manner. This makes the trust policy more dynamic because trusted CAs can be added or deleted from a network of secure servers without having to reconfigure each of the servers.

Related Information:

- For more information on LDAP, see "What is LDAP?".
- For more information on PKI, go to the IBM Security Library Web site at URL http://www.ibm.com/security/library/.

What is LDAP?

The HTTP Server supports the use of the Lightweight Directory Access Protocol (LDAP). LDAP is a protocol that provides access to an X.500 directory over a TCP or SSL connection.

LDAP enables you to store information in a directory service and query it in a database fashion. When you use X.500 directories and LDAP, any LDAP-enabled application can store information such as user authentication information once, and other applications using the LDAP server will recognize it. Using LDAP support also allows multiple HTTP Servers to share configuration information.

LDAP reduces required system resources by including only a functional subset of the original X.500 Directory Access Protocol (DAP). See the LDAP client toolkit documentation for a complete listing. The LDAP client toolkit is included when you install z/OS.

The HTTP Server LDAP support is extremely scalable. You have a choice of LDAP configurations including:

- A single LDAP server
- High availability configurations using multiple LDAP servers, for example, primary, secondary, and so on
- Different LDAP servers to be accessed for different requests. For example, requests may come in from two different IP addresses, and the Web server could contact a different LDAP server for each one.

Plug-ins can use the LDAP API directly. For information on writing LDAP APIs, seeChapter 20, "Accessing LDAP information with the LDAP API" on page 379.

This book assumes you have an existing X.500 directory service available, for example the IBM eNetwork X.500 directory. For more information, go to the URL:

http://www.ibm.com/software/network/directory/

X.500 overview

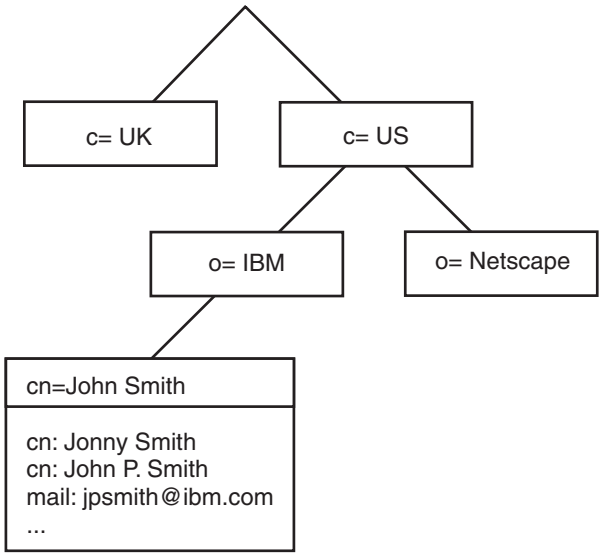
X.500 is a directory service with components that provide more efficient retrieval. LDAP uses two of these components: the information model, which determines the form and character, and the namespace, which allows information to be indexed and referenced.

The X.500 directory structure differs from others in the way the information is stored and retrieved. Information is associated with attributes. A query based on attributes is generated and sent to the LDAP server, and the server returns the respective values. LDAP uses a simple, string-based approach for representing directory entries.

An X.500 directory consists of *entries*, which are *typed* depending on the ObjectClass attribute. Each entry is composed of *attributes*. The ObjectClass attribute identifies the type of entry (for example, person or organization), which determines which attributes are required and which are optional.

Entries, which are arranged in a tree structure, may be divided among servers in geographical and organizational distribution. They are named according to their position within the distribution hierarchy by a distinguished name (DN).

Using the example directory tree structure, you can see how the DN and relative DN differ, as well as how they are used for identifying entries.

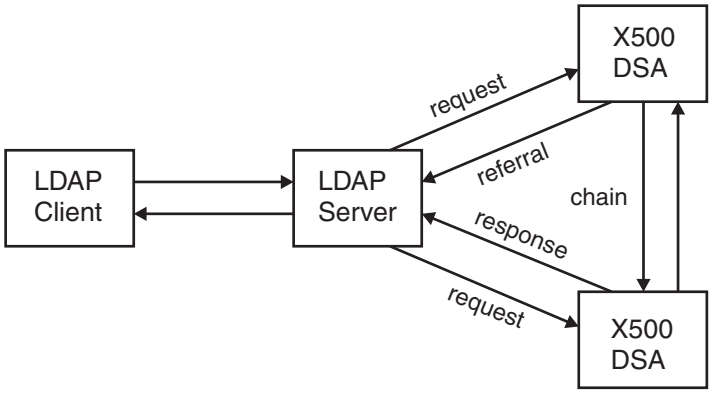


LDAP overview

Accessing an X.500 directory requires a certain protocol, for example Directory Access Protocol (DAP). However, DAP requires large amounts of system resources and support mechanisms to handle the complexity of the protocol. To allow desktop workstations to access the X.500, LDAP was introduced.

LDAP is client/server based, and some of the heavy resources required by DAP clients are handled by the LDAP server. An LDAP server can only return results or errors to the client, requiring little from the client. If an X.500 server is unable to

answer a client request, it must chain the request to another X.500 server. The server must complete the request or return an error to the LDAP server, which in turn passes the information to the client.



Related Information:

- For information on configuring LDAP support, see Chapter 15, "Retrieving Lightweight Directory Access Protocol information" on page 297.

Security options for the HTTP Server

The HTTP Server provides:
- "Options for setting up secure connections"
- "Options for protecting server resources" on page 163
- "SSL support for multiple IP addresses" on page 62
- "Support for specifying the encryption level to be used" on page 62

Options for setting up secure connections

The HTTP Server uses the following industry standards:

- Secure Sockets Layer (SSL) protocol for connection security
- Public key cryptography from RSA Data Security, Inc. for encryption and authentication
- X.500 and X.509 for certificate authentication and processing as part of an enterprise Public Key Infrastructure (PKI)

You do not have a secure network connection until you have set up SSL on the Web server. For examples, see "Examples: Setting up secure connections" on page 69.

Related Information:

- See the following sections for more information on concepts and terminology:
  - "What is SSL?" on page 58
  - "What is encryption?" on page 56
  - "What is authentication?" on page 57
  - "What is a Public Key Infrastructure?" on page 58

SSL support for multiple IP addresses

A secure network connection between a server and a browser requires an SSL session between them. During an SSL session with the server, a digital certificate (or digital ID) is sent to the browser. The certificate contains the host name of the server.

If your server is configured with multiple IP addresses, you can send different certificates from each IP address. If you have a different host name assigned to each IP address, you can configure SSL to send a different certificate for each host name. For example, if your server is configured to have the two IP addresses 125.25.116.87 and 125.25.116.89 with DNS entries of www.Mall.com and www.SuperShopper.com, respectively, each host name/IP address combination could have a separate key and certificate. Because the IP session connections for www.Mall.com and www.SuperShoper.com are coming in on different IP addresses for the requests, the server can serve the corresponding certificate for each host name.

Where there are multiple host name/IP address combinations, we recommend a separate certificate for each combination. Separate certificates prevent the browser pop-up warning that the host name of the request does not match the host name presented in the certificate during the initialization of the SSL session.

Related Information:

- For configuration instructions, see "Setting up SSL support for multiple IP addresses" on page 159.
- For more information on digital certificates, see "What is authentication?" on page 57.

Support for specifying the encryption level to be used

Related Information:

- For configuration instructions, see "Changing the default order of encryption levels used by the Web server" on page 160.
- For information on using S/390 cryptographic hardware, see "Hardware encryption" on page 65.

Certificate authorities supported by the HTTP Server

Your public key must be associated with a digitally signed certificate from a certificate authority (CA) who is designated as a trusted CA on your server.

There are two ways to obtain a certificate:

- "Buying a certificate from an external commercial CA"
- "Acting as your own CA" on page 63

Buying a certificate from an external commercial CA

You can buy a signed certificate by submitting a certificate request to a CA provider. z/OS System SSL supports the following external commercial certificate authorities:

VeriSign

For more information, go to the VeriSign Web site at URL http://www.verisign.com/.

Thawte
For more information, go to the Thawte Web site at URL http://www.thawte.com/.

By default, the following CAs are designated as trusted by z/OS System SSL:
- Integrion Certification Authority Root
- IBM World Registry Certification Authority
- Thawte Personal Premium CA
- Thawte Personal Freemail CA
- Thawte Personal Basic CA
- Thawte Premium Server CA
- Thawte Server CA
- VeriSign Test CA Root Certificate
- RSA Secure Server Certification Authority (from VeriSign)
- VeriSign Class 1 Public Primary Certification Authority
- VeriSign Class 2 Public Primary Certification Authority
- VeriSign Class 3 Public Primary Certification Authority
- VeriSign Class 4 Public Primary Certification Authority

For the most current information on default trusted CAs, see *z/OS System SSL Programming Guide and Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Note: If you are using certificate revocation lists (CRLs) for client authentication, you must purchase CA software from the IBM Registry and issue your own certificates. For more information, go to URL http://www.ibm.com/software/commerce/registry/

Acting as your own CA

If you act as your own CA, you create and sign your own CA certificate, and sign certificates for servers and clients that connect to your server. This is recommended only for test environments or private Web networks. Servers and clients must have browsers, such as Netscape Navigator or Microsoft Internet Explorer, to receive your CA certificate and designate you as a trusted CA.

To act as your own CA, you can use the z/OS System SSL gskkyman utility or RACF RACDCERT command to create and sign your certificates, or you can purchase certificate authority software from a CA provider. For an example using gskkyman, see "Examples: Setting up secure connections" on page 69. The HTTP Server supports the following CA software:

IBM Vault Registry
If you are using certificate revocation lists (CRLs) for client authentication, you must use CA software from IBM Vault Registry. For more information, go to URL: `http://www.ibm.com/software/commerce/registry/`.

Entrust Demo Certificates and Entrust WEBCA
For more information, go to URL: `http://www.entrust.com/`.

Netscape Certificate Server
For more information, go to URL: `http://www.netscape.com/`.

XCert For more information, go to URL: `http://www.xcert.com/`.

Any X.509-compliant certificate authority

Encryption support for the HTTP Server

The U.S. Government and foreign import rules regulate products used for encryption and prohibit their export unless their key size is strictly limited. This section summarizes the key sizes and Secure Sockets Layer (SSL) cipher specifications for the North American and export editions of the Web server. Customers in the United States and Canada can install the North American or export editions.

As U.S. export laws and foreign import rules are updated, the supported key lengths and cipher specifications are subject to change.

Note:
For the most current information on encryption support and supported key sizes, see the *z/OS System SSL Programming Guide and Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Supported public-private key sizes

Key size refers to the key lengths used for the public and private keys in an asymmetric key exchange. For an explanation of asymmetric keys, see "What is encryption?" on page 56.

North American edition (U.S. and Canada)

The North American edition of the server can:
- Generate keys from 512-1024 bits
- Encrypt data with symmetric keys from 40-168 bits
- Sign data with keys from 512-1024 bits
- Check signatures with keys from 512-1024 bits

International export edition

The International export edition of the server can do the following:
- Generate keys with a key size of 512 bits
- Encrypt data with symmetric keys from 40-56 bits
- Sign data with a key size of 512 bits
- Check signatures with a key size of 512 bits

Supported SSL cipher specifications

The SSL cipher specification tells you which data encryption algorithm and key size are used; for SSL V3, the hashing algorithm is included. Key size refers to the key length used for the keys in a symmetric key exchange. For an explanation of symmetric keys, see "What is encryption?" on page 56.

For example, cipher specification Triple-DES SHA uses the Triple-DES encryption algorithm and the SHA hashing algorithm. For more information on cipher specifications, go to URL:

`http://home.netscape.com/eng/ssl3/`

The HTTP Server uses a level of encryption supported by both the client and the server. You can use the SSLCipherSpec directive to change the default order of encryption levels used by the server. For instructions, see "Changing the default order of encryption levels used by the Web server" on page 160.

The HTTP Server supports the following SSL cipher specifications.The SSLCipherSpecs are listed in the order that is generally accepted by the industry as the order from strongest to weakest.

Note: For the most current information on supported cipher specifications, see the *z/OS System SSL Programming Guide and Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

North American edition (U.S. and Canada)

SSL V2:
27 Triple DES (192/168 bit)
21 RC4 (128 bit)
23 RC2 (128 bit)
26 DES (64/56 bit)
22 RC4 (40 bit)
24 RC2 (40 bit)

SSL V3:
3A Triple DES SHA (192/168 bit)
35 RC4 SHA (128 bit)
34 RC4 MD5 (128 bit)
39 DES SHA (64/56 bit)
33 RC4 MD5 (40 bit)
36 RC2 MD5 (40 bit)
32 NULL SHA
31 NULL MD5
30 NULL NULL

Note: Cipher specifications **32**, **31**, and **30** are standard SSL cipher specifications. However, they do not cause the data to be encrypted, and therefore do not provide data security. We only recommend them for debugging purposes.

International export edition

SSL V2:
22 RC4 (40 bit)
24 RC2 (40 bit)

SSL V3:
33 RC4 MD5 (40 bit)
36 RC2 MD5 (40 bit)
32 NULL SHA
31 NULL MD5
30 NULL NULL

Note: Cipher specifications **32**, **31**, and **30** are standard SSL cipher specifications. However, they do not cause the data to be encrypted, and therefore do not provide data security. We only recommend them for debugging purposes.

Hardware encryption

How hardware encryption affects Web server performance

You can use hardware encryption to improve performance of SSL sessions between the client and the server. By far, the biggest gain in performance for the Web server is in the SSL handshake. The handshake uses asymmetric keys and functions. The

Web server uses RSA technology to implement the asymmetric capability. When you implement SSL without hardware encryption, the asymmetric functions are much slower than symmetric functions. So when you implement hardware encryption with the Web server, make sure that you set up your asymmetric Master Keys properly using the Integrated Cryptographic Services Facility (ICSF) software so that you can take advantage of the performance boost. The asymmetric Master Keys are not the same as the Web server's RSA keys.

DES cipher specifications and Triple-DES cipher specifications use symmetric keys in order to handle data transmission. Data transmission may or may not be faster in hardware. Whether data transmission is faster in hardware or software depends on the size of the data stream. SSL should be sending relatively small streams of data, usually 4K bytes, or less. Smaller streams of data tend to be faster in software. Mid-range streams can be faster in hardware or software. Very large streams are faster in hardware.

Basic information needed to implement hardware encryption

- Insure that the cryptographic hardware is installed on your machines. You must have G3 or later machines or a Multiprise server in order to take advantage of hardware encryption. The hardware must be enabled and configured with diskettes from IBM. An IBM customer engineer (CE) will perform the enablement load. However, if you are using Processor Resource/Systems Manager (PR/SM), you must do additional hardware customization.. For more information on PRSM customization, see the *z/OS PR/SM Planning Guide* . To access this book on the Web, go to the z/OS Book Server Web site at URL:
  `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

  You must use the Cryptographic Coprocessor Feature (CCF) for cryptographic hardware. You can optionally use the PCI Cryptographic Coprocessor with CCF depending on your server model or your processor. The Web server uses RSA technology for the SSL handshake. The handshake is an asymmetric capability and uses RSA public-private key pairs. You can generate RSA keys in software or hardware.

  If you generate the RSA keys in software, you can use RACF commands or gskkyman. For more information on the generation of RSA keys in software, see "Examples: Setting up secure connections" on page 69.

  If you generate the RSA keys in hardware, you have a few options on how to do this. The PCI Cryptographic Coprocessor can generate RSA public-private key pairs within its secure hardware boundary. CCF without the PCI Cryptographic Coprocessor cannot. In this case CCF would need either a TKE workstation or a workstation with a special cryptographic adapter installed to generate RSA public-private key pairs. For more information on the use of cryptographic hardware with RSA technology, see the *z/OS ICSF Administrator's Guide* and the *z/OS ICSF System Programmer's Guide* . To access these books on the Web, go to the z/OS Book Server Web site at URL:
  `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`
- Configure the Integrated Cryptographic Services Facility (ICSF) software. ICSF is the software interface to the cryptographic hardware. Be sure to use ICSF to load the DES and PKA Master keys into the cryptographic hardware. For information on how to configure ICSF, see the *z/OS ICSF Administrator's Guide* and the *z/OS ICSF System Programmer's Guide* . To access these books on the Web, go to the z/OS Book Server Web site at URL:
  `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`
- Install the Web server on machines that contain cryptographic hardware. In addition, in order for the hardware to perform cryptographic operations on symmetric data streams you must do the following:

  - Configure the Web server to support DES or Triple-DES encryption levels. Triple-DES hardware encryption is available on G4 and later machines. For configuration instructions, see "Changing the default order of encryption levels used by the Web server" on page 160.
  - Set up your DES Master Key properly in ICSF.
- Define RACF commands in order to permit user IDs and the Web server's ID to profiles in the CSFSERV general resource class. CSFSERV controls the use of ICSF software. For information on how to define these RACF commands see "Permitting user IDs to CSFSERV for hardware encryption" on page 26. If you are using another security product, refer to that product's documentation for instructions.

How to check whether hardware encryption is used for Web server encryption

z/OS Version 1 Release 4 or later

Integrated Cryptographic Services Facility (ICSF) is the software interface to the cryptographic hardware. Use this checklist to determine if your Web server is working with hardware encryption.

- Verify that user IDs and the Web server ID have access to ICSF. For further information, see "Permitting user IDs to CSFSERV for hardware encryption" on page 26.
- Check that the ICSF started task is active.
- Do one or both of the following through the ICSF TSO panels to insure ICSF is working properly:
  - Check that ICSF has PKA Master Keys defined
  - Generate a PKA Master Key successfully.

For information on ICSF, see the *z/OS ICSF Administrator's Guide* . To access this book on the Web, go to the z/OS Book Server Web site at URL:
`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

z/OS Version 1 Release 3 or earlier

There are two messages that you can use to determine whether hardware encryption and your SSL environment are working. You can optionally turn on the following message which indicates whether hardware encryption is being used for Web server encryption:

```
Crypto Hardware {is|is not} being used by SSL.
```

This message will be written to stderr and to the -vv trace, if tracing is turned on. To enable this message, add the `GSK_SSL_HW_DETECT_MESSAGE` environment variable using either of the following methods:

- Edit the Web server start-up PROC, IMWEBSRV, and add:
  ```
  LEPARM='ENVAR("GSK_SSL_HW_DETECT_MESSAGE=1")'
  ```
- Edit the Web server httpd.envvars file and add:
  ```
  GSK_SSL_HW_DETECT_MESSAGE=1
  ```

You will see the following message immediately after the Crypto Hardware message described above.

```
retcode from skit_initialize=n
```

n indicates the return code for SSL initialization. A return code of 0 indicates that SSL initialized successfully.

Checklist for setting up a secure server

Use this checklist to help insure that your Secure Sockets Layer (SSL) environment initializes correctly and operates properly.

- Insure that you set up System SSL properly. For more information, see "Program control for z/OS System SSL" on page 25.
- Insure the paths to the security modules, `wwwus.so` for North American Security or `wwwexp.so` for export security, are specified on the LIBPATH environment variable in the `httpd.envvars` file.
- Insure that you have not set the following Web server environment variables:
  - -nosec
  - -sslmode off
- Verify that you have the following Web server directives defined correctly:

  SSLMode
  Insure that you code the SSLMode directive with option `On` and that the SSLMode directive appears only once in the `httpd.conf` file.

  KeyFile
  Insure that the server key database file that you specify on the KeyFile directive exists. Include either the absolute path to the key database file or the path to the key database file that is relative to the server root. The default server root is `/usr/lpp/internet/server_root`.

  If you have more than one KeyFile directive in your `httpd.conf` file, insure the server key database file that you want to use is specified on the last KeyFile directive. The Web server uses the last KeyFile directive specified.

  SSLCipherSpec
  Insure that you have at least one SSLCipherSpec directive coded in the configuration file and that any cipher specifications coded on the directive are supported and available.

  SSLServerCert
  Insure that the server certificate label that you specify on the SSLServerCert directive exists in the key ring coded on the KeyFile directive.
- Insure that neither the server certificate that the Web server accesses nor the Certificate Authority certificate that signed the server certificate is expired.
- If you create your key database file with the gskkyman utility, insure the permissions for the key database file and the path to the key database file allow the Web server ID read and write access.
- If you create your server certificate with the gskkyman utility, use the utility to store the encrypted database password in a stash file. Without a stash file, the Web server starts, but connections are not secure. The following list discusses how to resolve stash file problems.

  The Web server cannot find the stash file.
  Insure the following:
  - The stash file exists.
  - The stash file is in the same directory as the server key database file.
  - The prefix for the stash file and the server key database file are the same. For example, if the server key database file is `test.kdb`, the stash file is `test.sth`.
  - The suffix for the stash file is `.sth`.

The Web server cannot read the stash file.
Insure that the permissions for the stash file and the path to the stash file allow the Web server ID read access.

The Web server indicates that the stash file name is null.
The stash file name is null when the Web server cannot find the server key database file specified on the KeyFile directive. Resolve the problem by following the directions for the KeyFile directive described previously in this checklist.

The password in the stash file is wrong.
The gskkyman utility automatically creates the stash file name when you select the option to create a stash file. You should do nothing to alter the name or the location of the stash file. If you do, you can have a key database file with one password in it and the corresponding stash file with another password file in it. For example, you have a key database file in `/user/test.kdb` with a password of **testpass**. The stash file in `/user/test.sth` contains the encrypted **testpass** password. You copy the stash file to `/user2/test.sth`. In the subdirectory `user2`, there is a key database file also called `test.kdb`. The `/user2/test.kdb` file has a password of **test2**. You specify `KeyFile /user2/test.kdb` in the `httpd.conf` file. When you start the Web server in secure mode, you receive an indication that the password in the stash file is incorrect. The password is incorrect because the `/user2/test.sth` file contains the password **testpass**, but the `/user2/test.kdb` file requires the **test2** password.

The password in the stash file has expired.
This is the password that you entered when you created the server key database. You can either set the password to expire in a specific number of days, or set the password so it never expires.

Examples: Setting up secure connections

Important notes

gskkyman and RACDCERT examples::

You can use both the z/OS Secure Sockets Layer (SSL) gskkyman utility and the Resource Access Control Facility (RACF) RACDCERT command to create, process, and manage files needed for secure connections. This section includes three sets of step-by-step examples. One set shows you how to create secure connections that use the gskkyman utility on z/OS Version 1 Release 3 and earlier releases. Another set shows you how to create secure connections that use the gskkyman utility on z/OS Version 1 Release 4 and later releases. A third set shows you how to create secure connections that use the RACDCERT command. The examples were designed so that the corresponding gskkyman and RACDCERT examples use the same inputs whenever possible. Updates to these examples are available in the Portable Document Format (PDF) file of the Web server manual on the Web site at URL `http://www.ibm.com/software/webservers/httpservers/doc53.html`.

If you plan to use the gskkyman utility to create keys and certificates, be aware that the menus changed between z/OS Version 1 Release 3 and z/OS Version 1 Release 4. Choose the set of examples that go with your z/OS release. If unsure whether you are on z/OS Version 1 Release 4 or a later release, you can quickly tell by viewing the main menu of the gskkyman utility.

For z/OS Version 1 Release 4 and later, the main menu appears as follows:

```
       Database Menu

   1 - Create new database
   2 - Open database
   3 - Change database password
   4 - Change database record length
   5 - Delete database

   0 - Exit program

Enter option number:
```

For z/OS Version 1 Release 3 and earlier, the main menu appears as follows:

```
            IBM Key Management Utility

Choose one of the following options to proceed.

   1  - Create new key database
   2  - Open key database
   3  - Change database password

   0  - Exit program

Enter your option number:
```

z/OS System SSL notes:

1. Before using the System SSL gskkyman examples, ensure that you have set up your system environment to support System SSL and the gskkyman utility. For information on setup requirements and detailed information on gskkyman, see the *z/OS System SSL Programming Guide and Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.
2. The gskkyman utility is not shipped as part of the HTTP Server. Contact z/OS System SSL Support for assistance with System SSL or gskkyman problems.

z/OS RACF notes:

1. For complete syntax on the RACF RACDCERT command, see the *z/OS Security Server (RACF): Command Language Reference*. For other examples that use the RACF RACDCERT command, see the *z/OS System SSL Programming Guide and Reference*. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.
2. If you are using RACF to create, process, and manage the files needed for secure connections, you must specify the SAF option on the Web server KeyFile directive, for example, `KeyFile` *`key_file_name`* `SAF`.
3. For assistance with RACF problems, contact z/OS Security Server support.

Which example should I use?

"Example 1A for gskkyman: Setting up secure connections using a self-signed server certificate (z/OS Version 1 Release 4 or later)" on page 75

Use this example for z/OS Version 1 Release 4 or later releases. Using a self-signed server certificate is recommended only for test environments. This method gives you a quick way to test your SSL setup and internal applications with a small number of trusted clients.

"Example 2A for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 4 or later)" on page 78

page 78

Use this example for z/OS Version 1 Release 4 or later releases. Using an external commercial CA, such as VeriSign or Thawte, is recommended for Internet connections and e-business. This method provides the highest level of security. You can optionally use client authentication to verify the identity of those accessing your server.

"Example 3A for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 4 or later)" on page 84

Use this example for z/OS Version 1 Release 4 or later releases. Acting as your own CA is recommended only for test environments and private intranets. With this method, you set up your own CA and sign certificates. You can optionally use client authentication to verify the identity of those accessing your server.

"Example 4A for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 4 or later)" on page 90

Use this example for z/OS Version 1 Release 4 or later releases.

This example shows how to set up client authentication using client certificates. Before using this example, set up secure connections using either of the following examples:

- "Example 2A for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 4 or later)" on page 78
- "Example 3A for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 4 or later)" on page 84

"Example 1 for gskkyman: Setting up secure connections using a self-signed server certificate (z/OS Version 1 Release 3 or earlier)" on page 98

Use this example for z/OS Version 1 Release 3 or earlier releases. Using a self-signed server certificate is recommended only for test environments. This method gives you a quick way to test your SSL setup and internal applications with a small number of trusted clients.

"Example 2 for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 3 or earlier)" on page 101

Use this example for z/OS Version 1 Release 3 or earlier releases. Using an external commercial CA, such as VeriSign or Thawte, is recommended for Internet connections and e-business. This method provides the highest level of security. You can optionally use client authentication to verify the identity of those accessing your server.

"Example 3 for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 3 or earlier)" on page 107

Use this example for z/OS Version 1 Release 3 or earlier releases. Acting as your own CA is recommended only for test environments and private intranets. With this method, you set up your own CA and sign certificates. You can optionally use client authentication to verify the identity of those accessing your server.

"Example 4 for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 3 or earlier)" on page 113

Use this example for z/OS Version 1 Release 3 or earlier releases. This example shows how to set up client authentication using client certificates. Before using this example, set up secure connections using either of the following examples:

- "Example 2 for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 3 or earlier)" on page 101
- "Example 3 for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 3 or earlier)" on page 107

"Example 5 for RACDCERT: Setting up secure connections using a self-signed server certificate" on page 120

Using a self-signed server certificate is recommended only for test environments. This method gives you a quick way to test your SSL setup and internal applications with a small number of trusted clients.

"Example 6 for RACDCERT: Setting up secure connections using certificates signed by an external commercial CA" on page 124

Using an external commercial CA, such as VeriSign or Thawte, is recommended for Internet connections and e-business. This method provides the highest level of security. You can optionally use client authentication to verify the identity of those accessing your server.

"Example 7 for RACDCERT: Setting up secure connections using certificates signed by an internal CA" on page 130

Acting as your own CA is recommended only for test environments and private intranets. With this method, you set up your own CA and sign certificates. You can optionally use client authentication to verify the identity of those accessing your server.

"Example 8 for RACDCERT: Setting up client authentication using client certificates signed by an internal CA" on page 135

This example shows how to set up client authentication when acting as your own CA. Before using this example, you must set up secure connections using either of the following examples:

- "Example 6 for RACDCERT: Setting up secure connections using certificates signed by an external commercial CA" on page 124
- "Example 7 for RACDCERT: Setting up secure connections using certificates signed by an internal CA" on page 130

"Example 9 for RACDCERT: Setting up client authentication using client certificates signed by an external CA" on page 140

This example shows how to set up client authentication when using an external CA. Before using this example, you must set up secure connections using either of the following examples:

- "Example 6 for RACDCERT: Setting up secure connections using certificates signed by an external commercial CA" on page 124
- "Example 7 for RACDCERT: Setting up secure connections using certificates signed by an internal CA" on page 130

"Example 10A: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1 Release 4 or later)" on page 145

Use this example for z/OS Version 1 Release 4 or later releases. If you have set up your secure environment using either gskkyman or IKEYMAN, you can migrate the environment to a RACF database. RACF provides a more secure environment for your certificates and keys because they are stored in a database rather than in a z/OS UNIX System Services file.

"Example 10: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1

Release 3 or earlier)" on page 151
Use this example for z/OS Version 1 Release 3 or earlier releases. If you have set up your secure environment using either gskkyman or IKEYMAN, you can migrate the environment to a RACF database. RACF provides a more secure environment for your certificates and keys because they are stored in a database rather than in a z/OS UNIX System Services file.

Terms used in these examples

This section defines the main terms used in the examples. For more detailed information, refer to the books and resources in "Related documentation and URLs" on page 159.

CA (certificate authority or certification authority)
A CA is a trusted third-party who issues server and client certificates used in secure SSL connections. The CA authenticates the certificate owner's identity and the services that the owner is authorized to use, issues new certificates, renews existing certificates, and revokes certificates belonging to users who are no longer authorized to use them. You can use an external commercial CA such as VeriSign or Thawte, or you can act as your own CA.

CA certificate
If you use a CA, that CA must be designated as a trusted CA in your browser. If you are acting as your own CA, you must create your own CA certificate, store the certificate on your server, and send the certificate to all servers and clients that will connect to your server. If you use an external commercial CA, servers and clients may not have to obtain and store the CA's certificate; by default, many CAs are already designated as trusted in browsers such as Netscape and Microsoft Internet Explorer.

CA key database (also called a CA key ring)
If you act as your own CA and use the gskkyman utility, create a CA key database in addition to a server key database. The CA key database contains your CA certificate and is used only to sign server and client certificates. The server key database is required for SSL operations.

certificate
A certificate (or digital certificate) is a digital document that associates a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. A certificate can be signed by the originator of the certificate (self-signed) or issued by a trusted third-party called a CA. A certificate contains:
- Information about the certificate owner's identity called the owner's Distinguished Name (DN)
- The public key of the person being certified
- The CA's Distinguished Name (DN)
- The signature of the CA

client certificate (also called user or browser certificate)
If you are using SSL V3, you can optionally authenticate clients using client certificates. A client certificate verifies the client's identity to the server and is signed by a trusted third-party called a CA. For a client certificate to be used, the CA certificate must be stored in the client's browser and designated as a trusted CA.

client certificate request
A client certificate request contains the client's public-private key pair and unsigned certificate. The client certificate request is signed by a CA.

key database (also called a key ring)
If you are installing and maintaining digital certificates and key databases using gskkyman, a key database is a file that contains public keys, private keys, trusted CAs, and certificates. Also see, CA key database and server key database.

key ring
If you are installing and maintaining digital certificates and key rings in RACF, then public keys, private keys, trusted CAs, and certificates are stored in a key ring in the RACF database. Key ring names become names of RACF profiles in the DIGTRING class once the ring is added to RACF. Also see, server key ring.

secure connections
To use the Secure Sockets Layer (SSL) protocol for secure connections, both server and client must have SSL protocol software. To set up a secure connection, the SSL protocol initializes a secure session that includes the following basic steps, referred to as the SSL handshake:

1. The client sends a message which includes the encryption levels it supports.
2. The server responds and sends its server certificate, which includes the server public key, and the encryption levels it supports. If the server is using SSL V3 and client authentication is set up, it can include a request for a client certificate to verify the identity of the client.
3. The client selects an encryption level supported by both, then sends data encrypted with the server's public key. If client authentication is being used, the client must also send its client certificate to the server.
4. When the server receives the encrypted data, it decrypts the data with its private key; if sent, the server also validates the client certificate, then decides whether to proceed with the communication.

self-signed certificate
A self-signed certificate is signed by the creator of the certificate and not by a trusted third-party (CA). Using self-signed certificates is recommended only for test environments.

server certificate (also called server ID)
A server certificate contains the public-private key pair for the server and a certificate. A server certificate can be self-signed or signed by a CA.

server certificate request
A server certificate request contains the server's public-private key pair and unsigned certificate. The server certificate request is signed by a CA.

server key database (also called a server key ring)
If you are installing and maintaining digital certificates and key databases using gskkyman, a server key database is required for SSL operations. The fully qualified path and name of this database must appear on the KeyFile directive in the server configuration file. You do not need to include the CA key database on the KeyFile directive. The CA key database is used for signing server and client certificates when you are acting as your own CA; it is not required for SSL operations.

server key ring
If you are installing and maintaining digital certificates and key rings in RACF, a server key ring is required for SSL operations. The name of this ring along with the key word SAF must appear on the KeyFile directive in the server configuration file.

Example 1A for gskkyman: Setting up secure connections using a self-signed server certificate (z/OS Version 1 Release 4 or later)

Note: The gskkyman utility changed in z/OS Version 1 Release 4. Use this example only if your system is at Version 1 Release 4 or a later release.

This example is a quick start method for setting up SSL on your server.

Note: Using a self-signed server certificate is recommended only for test environments. This method gives you a quick way to test your SSL setup and internal applications with a small number of trusted clients.

Note: If you run multiple Web servers, make sure that you store the server, Certificate Authority (CA), and client key databases for a particular Web server in a separate set of files from key databases belonging to other Web servers. Otherwise, if one of the files becomes corrupted, you lose key databases for multiple Web servers. SSL does not initialize for any of the Web servers whose server key databases are stored in a corrupted file.

System SSL notes:

You must use System SSL to establish secure connections. To use System SSL with the Web server, the System SSL load library must exist in linklist and must be under program control. If you have not already done so:

- Add the load library to the linklist.
- Turn on program control for the library by issuing the following RACF commands from a user ID that has the proper authority:

```
RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

  If turning on program control for the first time, use the RDEFINE command instead of the RALTER command.

For an explanation of terms, refer to “Terms used in these examples” on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in “Related documentation and URLs” on page 159.

Summary of tasks

In this example, you perform the following main tasks:

1. Create the server key database
2. Create your self-signed server certificate
3. Register the server key database with the Web server
4. Use server security defaults
5. Verify that you can establish a secure connection with the server

Step 1. Create the server key database

A server key database is required for each server that supports secure SSL connections. You must specify the fully qualified path and name of the server key database on the KeyFile directive in the server configuration file, `httpd.conf`.

In this step, you create the following files:

- `serva.kdb` is the server key database used for SSL operations.

- `serva.rdb` is the server request database which contains the server public-private key pair. When the gskkyman utility generates this file, it uses the same name you specify for the server key database.
- `serva.sth` is the stash file containing the encrypted password for the server key database. This file is required for secure SSL connections and must be in the same directory as the server key database. If the Web server cannot find the stash file, the server starts with a non-secured connection. When gskkyman generates this file, the utility uses the same name you specify for the server key database.

To create your server key database:

1. Change to the path of the new server key database location by entering `cd` *`path`*. In this example, *path* is `/u/serva`.

   Note: For added security of the stash file, set the permissions for the subdirectory to 700.
2. Enter `gskkyman` to start the utility.
3. Enter **1** to create a new server key database.
4. Enter a unique name for your server key database. In this example, `serva.kdb` is used.
5. Enter a password using the guidelines in "Key database password" on page 158.
6. Enter the password again for verification.
7. Specify the number of days until the password expires, or never allow the password to expire.
8. Enter the database record length, or accept the default record length of 2500. The gskkyman utility displays a message confirming that the database is created.
9. Press **Enter** to continue working with the server key database.
10. Enter **10** to store the encrypted database password in a stash file. A message displays confirming that the stash file is created. The name of the stash file is the same as the database name, for example, `serva.sth`.

    Note: You must store the encrypted password for your server key database in a stash file. This file is required for secure SSL connections and must be in the same directory as the server key database. If this file is not created, the server starts but the connection is not secure.
11. Press **Enter** to continue working with the server key database.

Step 2. Create your self-signed server certificate

In this step, you create your public-private key pair and self-signed server certificate.

To create your self-signed server certificate:

1. Enter **6** to create a self-signed server certificate.
2. Enter **4**, **5**, or **6** to select one of the end user certificate types along with its key size. A smaller key size impacts performance less because it requires fewer resources to encrypt and decrypt than a larger key size. However, you need to determine the appropriate key size for your security needs. For more information on encryption and supported key sizes, see "Encryption support for the HTTP Server" on page 64.

Note: An end user certificate can either be a server certificate or a client certificate. In this step the end user certificate is a server certificate.

3. Enter a name or label to use for identifying the key and certificate in the database, for example, **Certificate for serva**.
4. Complete the fields that generate the Distinguished Name (DN):
   - Enter the common name or Domain Name of the server, for example, **serva.raleigh.ibm.com**.
   - Optionally enter an organization unit, for example, **IBM ID z/OS**.
   - Enter an organization name, for example, **IBM ID Raleigh**.
   - Optionally enter a city or locality, for example, **Raleigh**.
   - Optionally enter a state or province, for example, **North Carolina**
   - Enter a country code, for example, **US**. Only two characters are allowed.
5. Indicate the number of days that the certificate is valid, or accept the default of 365 days.
6. Press **Enter** to continue, after the gskkyman utility successfully creates your certificate.
7. Indicate that the certificate key is the default key in the database by doing the following:
   - Enter **1** to mange your keys and certificates.
   - Select **Certificate for serva**.
   - Enter **3** to select the key as the default.

   Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.
8. Press **Enter** to continue, once the gskkyman utility sets the default key.
9. Verify that your self-signed server certificate was created by doing the following:
   a. Enter **1** to show certificate information. The subject name and the issuer should match.
   b. Enter **0** to return to the menu.
   c. Enter **2** to show key information. The default key should indicate **Yes**.
   d. Press **Enter**, then enter **0** to exit the gskkyman utility.

Step 3. Register the server key database with the Web server

To register the server key database in this example with the Web server, add the following KeyFile directive to your server configuration file, `httpd.conf`:

```
KeyFile  /u/serva/serva.kdb
```

Step 4. Use server security defaults

In this example, the following security default settings are used:

SSLMode
: Support for SSL connections is set **on**.

SSLPort
: Port 443 is the default port used for SSL.

SSLCipherSpec
: The server uses an encryption level, supported by both the client and the server.

Step 5. Verify that you can establish a secure connection with the server

To verify that you can establish a secure connection with your server, start your Web server. Then start a browser and enter the following URL:

```
https://your_server_name:SSL_port_number
```

The URL for this example is:

```
https://serva.raleigh.ibm.com:443
```

Example 2A for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 4 or later)

Note: The gskkyman utility changed in z/OS Version 1 Release 4. Use this example only if your system is at Version 1 Release 4 or a later release.

This example includes steps for using an external commercial CA to sign your server certificate.

Note: Using an external commercial CA, such as VeriSign or Thawte, is recommended for Internet connections and e-business. This method provides the highest level of security. You can optionally use client authentication to verify the identity of those accessing your server.

Note: If you run multiple Web servers, make sure that you store the server, Certificate Authority (CA), and client key databases for a particular Web server in a separate set of files from key databases belonging to other Web servers. Otherwise, if one of the files becomes corrupted, you lose key databases for multiple Web servers. SSL does not initialize for any of the Web servers whose server key databases are stored in a corrupted file.

System SSL notes:

You must use System SSL to establish secure connections. To use System SSL with the Web server, the System SSL load library must exist in linklist and must be under program control. If you have not already done so:

- Add the load library to the linklist.
- Turn on program control for the library by issuing the following RACF commands from a user ID that has the proper authority:

  ```
  RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
  SETROPTS WHEN(PROGRAM) REFRESH
  ```

  If turning on program control for the first time, use the RDEFINE command instead of the RALTER command.

For an explanation of terms, refer to “Terms used in these examples” on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in “Related documentation and URLs” on page 159.

Summary of tasks

In this example, you perform the following main tasks:

1. Set up the server key database
2. Send your server certificate request to an external commercial CA
3. Add the CA to the list of trusted CAs on the server, if necessary.
4. Receive your signed server certificate into the server key database

5. Use server security defaults
6. Verify that you can establish a secure connection with the server
7. Optionally, set up client authentication using client certificates

Step 1. Set up the server key database

A server key database is required for each server that supports secure SSL connections. You must specify the fully qualified path and name of the server key database on the KeyFile directive in the server configuration file, `httpd.conf`.

In this step, you create the following files:

- `server1.kdb` is the server key database used for SSL operations.
- `server1.rdb` is the server request database which contains the server public-private key pair. When the gskkyman utility generates this file, it uses the same name you specify for the server key database.
- `server1.sth` is the stash file containing the encrypted password for the server key database. This file is required for secure SSL connections and must be in the same directory as the server key database. If the Web server cannot find the stash file, the server starts with a non-secured connection. When the gskkyman utility generates this file, the utility uses the same name you specify for the server key database.
- `certreq.arm` is an ASCII file which contains your server public-private key pair and unsigned certificate. This is the default name.

Creating your server key database: To create your server key database:

1. Change to the path of the new server key database location by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.

   Note: For added security of the stash file, set the permissions for the subdirectory to 700.
2. Enter `gskkyman` to start the utility.
3. Enter **1** to create a new server key database.
4. Enter a unique name for your server key database. In this example, `server1.kdb` is used.
5. Enter a password using the guidelines in "Key database password" on page 158.
6. Enter the password again for verification.
7. Specify the number of days until the password expires, or never allow the password to expire.
8. Enter the database record length, or accept the default record length of 2500. The gskkyman utility displays a message confirming that the database is created.
9. Press **Enter** to continue working with the server key database.
10. Enter **10** to store the encrypted database password in a stash file. A message displays confirming that the stash file is created. The name of the stash file is the same as the database name, for example, `server1.sth`.

    Note: You must store the encrypted password for your server key database in a stash file. This file is required for secure SSL connections and must be in the same directory as the server key database. If this file is not created, the server starts but the connection is not secure.
11. Press **Enter** to continue working with the server key database.

Creating your server certificate request: In this step, you create your server certificate request which includes your public-private key pair and unsigned certificate. To create your server certificate request:

1. Enter **4** to create a new certificate request.
2. Select one of the certificate types along with its key size. A smaller key size impacts performance less because it requires fewer resources to encrypt and decrypt than a larger key size. However, you need to determine the appropriate key size for your security needs. For more information on encryption and supported key sizes, see "Encryption support for the HTTP Server" on page 64.
3. Enter the request file name, `certreq.arm`. For information on certificate file types, see "Certificate file types generated by the gskkyman utility" on page 157.
4. Enter a name or label to use for identifying the key and certificate in the database, for example, **Certificate for server1**. Avoid using the word `request` when specifying labels.
5. Complete the fields that generate the Distinguished Name (DN):
   - Enter the common name or Domain Name of the server, for example, **server1.raleigh.ibm.com**.
   - Optionally enter an organization unit, for example, **IBM ID z/OS** .
   - Enter an organization name, for example, **IBM ID Raleigh**.
   - Optionally enter a city or locality, for example, **Raleigh**.
   - Optionally enter a state or province, for example, **North Carolina**
   - Enter a country code, for example, **US**. Only two characters are premitted.

   A message displays confirming that the request has completed successfully. The message indicates that the gskkyman facility created the `server1.rdb` and `certreq.arm` files.

   Note: Do not attempt to edit or move the `server1.rdb` file. If this file is not present or is corrupted when you attempt to receive your signed server certificate into the server key database, you have to resubmit your certificate request.
6. Press **Enter** to continue, then enter **0** to exit the gskkyman utility.

Step 2. Send your server certificate request to an external commercial certificate authority

After you create your server certificate request, you send that request to a certificate authority (CA) to sign. First transfer the file containing the server certificate request to your workstation. Then follow the instructions of the CA for sending and receiving your certificate request.

By default, the following CAs are designated as trusted on the Web server:

VeriSign

For more information on the following certificates, go to the VeriSign Web site at URL: `http://www.verisign.com/`.

- RSA Secure Server Certification Authority
- VeriSign Class 1 Public Primary Certification Authority
- VeriSign Class 2 Public Primary Certification Authority
- VeriSign Class 3 Public Primary Certification Authority
- VeriSign Class 1 Individual Subscriber- Persona Not Validated
- VeriSign Class 2 Individual Subscriber- Persona Not Validated
- VeriSign Class 3 Individual Subscriber- Persona Not Validated

Thawte

For more information on the following certificates, go to the Thawte Web site at URL: `http://www.thawte.com/`.

- Thawte Personal Basic Certificate Authority
- Thawte Personal Freemail Certificate Authority
- Thawte Personal Premium Certificate Authority
- Thawte Premium Server Certificate Authority
- Thawte Server Certificate Authority

Transferring the server certificate request to your workstation: This example shows how to use the File Transfer Protocol (FTP) command to transfer the `certreq.arm` file containing the server certificate request to your workstation. Perform the following steps on the workstation:

1. Enter the **FTP** command and either the host name or the IP address of the server. For example, enter `ftp server1.raleigh.ibm.com`.
2. Enter your user ID and password when prompted.
3. Change to the directory where you put the file containing the server certificate request, `certreq.arm`. For example, enter `cd /ibm/security/user1`.
4. Transfer the file to the workstation in ASCII format by entering `get certreq.arm`.
5. Type `quit` to exit.

Step 3. If necessary, add the CA to the list of trusted CAs on the server

If you send your certificate request to a CA that is already a trusted CA on your server, you can skip this step. In this step you first check if the CA is trusted on your server. If it is not, contact the CA to obtain their CA certificate. Transfer the CA certificate from your workstation to the host. Then import the CA certificate into your server key database and mark it as **Trusted**.

Checking the list of trusted CAs: To check the current list of trusted CAs on your server:

1. Change to the path where the server key database is located by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.
2. Enter **gskkyman** to start the utility.
3. Enter **2** to open the server key database.
4. Enter the server key database name, `server1.kdb`.
5. Enter the database password.
6. Enter **2** to list all trusted CAs. If the CA is not in the list, you must import the CA certificate into your server key database before you receive your signed server certificate.
7. Press **Enter** to continue working with the server key database.

Transferring the CA certificate from your workstation to the host: This example shows how to use the File Transfer Protocol (FTP) command to transfer the file containing your commercial CA certificate to the host. In this example the file name is `cert.arm`. Perform the following steps on the workstation:

1. Enter the **FTP** command and either the host name or the IP address of the server, for example, **ftp server1.raleigh.ibm.com**.
2. Enter your user ID and password, when prompted.
3. Change to the directory where you are putting the file containing the CA certificate, `cert.arm`, for example, **cd /ibm/security/user1**.
4. Transfer the file to the host in ASCII format by entering **put cert.arm**.
5. Type **quit** to exit.

Importing the CA certificate into your server key database: To designate a CA as trusted on your server, store the certificate in your server key database:

1. Enter **7** to import a certificate.
2. Enter the CA certificate file name, `cert.arm`.
3. Enter the CA label, **CA certificate for server1**.
4. Press **Enter** to return to the Key Management menu, after your request completes successfully.
5. Verify that your CA is now listed as a trusted CA on your server:
   a. Enter **2** to manage certificates.
   b. Find and select your certificate label in the list, for example, **CA certificate for server1**
   c. Enter **1** to show CA certificate information. **Trusted** should indicate **Yes**.
6. Enter **0**, press **Enter**, and then enter **0** to exit the gskkyman utility.

Step 4. Receive your signed server certificate into the server key database

In this step, you create the `servcert.arm` file. This file is in ASCII format. It contains the server public-private key pair and the server certificate signed by the external commercial CA.

Example of a signed server certificate: The certificate you receive should contain `BEGIN CERTIFICATE` and `END CERTIFICATE` lines with data between as shown in the following example. If your certificate contains additional information before `BEGIN CERTIFICATE` or after `END CERTIFICATE`, do not include the extraneous information in the certificate file you create in this step.


```
-----BEGIN CERTIFICATE-----
 MIIB0DCCATkCBDV8PgswDQYJKoZIhvcNAQECBQAwcjELMAkGA1UEBhMCVVMxDTAL
 BgNVBAgTBE4uQy4xDDAKBgNVBAcTA1JUUDEMMAoGA1UEChMDSUJNMRcwFQYDVQQL
 Ew5XZWJzZXJ2ZXIgVGVzdDEfMB0GA1UEAxMWbXZzMTY3LnJhbGVpZ2guaWJtLmNv
 bTAaFws5ODA2MDgxOTM5WhcLOTkwNjA4MTkzOVowNDELMAkGA1UEBhMCVVMxDDAK
 BgNVBAoTA0lCTTEXMBUGA1UEAxMOcGtjczEwLmlibS5jb20wXDANBgkqhkiG9w0B
 AQEFAANLADBIAkEA1IYGldVmnKAI8hJQGT074oXTD0Tb+jFN8wkPqc+DVhYix1fj
 h/sbiuDZF66BMh5hnHfJr75633CgjW10EpID0wIDAQABMA0GCSqGSIb3DQEBAgUA
 A4GBAF1KVppAM7Gh2F9BBIY/jPMFlRp8+HAAVkK29Q4DxeF2FrTzQutKmO8duCWv
 xnJo4pgl5Uj29DSAsrX8mULfczyuZwVVXiCGnhN03pYj8bbQjo0edqQ7hYsRl3P4
 C72I+yRwtWUukfVgwdo0mWXyEclx7eT5jsW4weVEqWvuht8j
-----END CERTIFICATE-----
```


Before you can receive the signed certificate into your server key database, create a certificate file and copy your signed server certificate into that file. This example assumes that you create the certificate file on your workstation and that you name it `servcert.arm`. Include the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines and all data in between as shown in the example. Remove any extraneous information that the CA included in the file.

Transferring the signed server certificate from your workstation to the host: This example shows how to use the File Transfer Protocol (FTP) command to transfer the file containing the signed server certificate to the host. In this example the file name is `servcert.arm`. Perform the following steps on the workstation:

1. Enter the **FTP** command and either the host name or the IP address of the server. For example, enter `ftp server1.raleigh.ibm.com`.
2. Enter your user ID and password when prompted.
3. Change to the directory where you are putting the file containing the signed server certificate, `servcert.arm`. For example, enter `cd /ibm/security/user1`.

4. Transfer the file to the host in ASCII format by entering `put servcert.arm`.
5. Type **quit** to exit.

To receive your signed certificate into the server key database:

1. Change to the path where the server key database is located by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.
2. Enter `gskkyman` to start the utility.
3. Enter **2** to open your server key database.
4. Enter the database name, `server1.kdb`.
5. Enter the database password.
6. Enter **5** to receive your server certificate.
7. Enter the name of your signed server certificate. In this example, that name is `servcert.arm`.
8. Press Enter to continue, after the gskkyman utility successfully receives your certificate.
9. Indicate that the certificate key is the default key in the database by doing the following:
   - Enter **1** to manage your keys and certificates.
   - Select **Certificate for server1**.
   - Enter **3** to select the key as the default.

   Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.
10. Press **Enter** to continue.
11. Enter **2** to show key information. The default key should indicate **Yes**.
12. Press **Enter**, then enter **0** to exit the gskkyman utility.

Step 5. Register the server key database with the server

To register your server key database in this example with the Web server, add the following KeyFile directive to your server configuration file, `httpd.conf`:

```
KeyFile  /ibm/security/user1/server1.kdb
```

Step 6. Use server security defaults

In this example, the following security default settings are used:

SSLMode
: Support for SSL connections is set **on**.

SSLPort
: Port 443 is the default port used for SSL.

SSLCipherSpec
: The server uses an encryption level, supported by both the client and the server.

You can change the Web server security defaults, by editing the appropriate security directives in the configuration file.

Step 7. Verify that you can establish a secure connection with the server

To verify that you can establish a secure connection with your server, start your Web server. Then start a browser and enter the following URL:

```
https://your_server_name:SSL_port_number
```

The URL for this example is:

```
https://serva.raleigh.ibm.com:443
```

Step 8. Optionally, set up client authentication using client certificates

For instructions, see "Example 4A for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 4 or later)" on page 90.

Example 3A for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 4 or later)

Note: The gskkyman utility changed in z/OS Version 1 Release 4. Use this example only if your system is at Version 1 Release 4 or a later release.

This example includes steps for setting up and administering your own certificate authority.

Note: Acting as your own CA is recommended only for test environments and private intranets. With this method, you set up your own CA and sign certificates. You can optionally use client authentication to verify the identity of those accessing your server.

Note: If you run multiple Web servers, make sure that you store the server, Certificate Authority (CA), and client key databases for a particular Web server in a separate set of files from key databases belonging to other Web servers. Otherwise, if one of the files becomes corrupted, you lose key databases for multiple Web servers. SSL does not initialize for any of the Web servers whose server key databases are stored in a corrupted file.

System SSL notes:

You must use System SSL to establish secure connections. To use System SSL with the Web server, the System SSL load library must exist in linklist and must be under program control. If you have not already done so:

- Add the load library to the linklist.
- Turn on program control for the library by issuing the following RACF commands from a user ID that has the proper authority:

  ```
  RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
  SETROPTS WHEN(PROGRAM) REFRESH
  ```

  If turning on program control for the first time, use the RDEFINE command instead of the RALTER command.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Summary of tasks

In this example, you perform the following main tasks:

1. Set up the CA key database
2. Set up the server key database
3. Sign your server certificate
4. Receive your signed server certificate into the server key database
5. Add your CA name to the list of trusted CAs for your browser
6. Register the server key database with the Web server
7. Use server security defaults

8. Verify that you can establish a secure connection with the server
9. Optionally, set up client authentication using client certificates

Step 1. Set up the CA key database

In this step, you create the following files:

- `cakey.kdb` is the CA key database.
- `cert.arm` is an ASCII file which contains your server public-private key pair and self-signed CA certificate. This is the default name.

Creating your CA key database:

1. Change to the path of the new CA key database location by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.
2. Enter `gskkyman` to start the utility.
3. Enter **1** to create a new CA key database.
4. Enter a unique name for your CA key database name. In this example, `cakey.kdb` is used.
5. Enter a password using the guidelines in "Key database password" on page 158.
6. Enter the password again for verification.
7. Specify the number of days until the password expires, or never allow the password to expire.
8. Enter the database record length, or accept the default record length of 2500. The gskkyman utility displays a message confirming that the database is created.
9. Press **Enter** to continue working with the CA key database.

Creating your self-signed CA certificate:

1. Enter **6** to create a self-signed CA certificate.
2. Enter **1**, **2**, or **3** to select one of the CA certificate types along with its key size. A smaller key size impacts performance less because it requires fewer resources to encrypt and decrypt than a larger key size. However, you need to determine the appropriate key size for your security needs. For more information on encryption and supported key sizes, see "Encryption support for the HTTP Server" on page 64.
3. Enter a name or label used to identify the key and certificate in the database, for example, **CA certificate for server1**.
4. Complete the fields that generate the Distinguished Name (DN):
   - Enter a common name for your CA, for example, **IBM Raleigh z/OS ID server CA**.
   - Optionally enter an organization unit, for example, **IBM ID z/OS** .
   - Enter an organization name, for example, **IBM ID Raleigh**.
   - Optionally enter a city or locality, for example, **Raleigh**.
   - Optionally enter a state or province, for example, **North Carolina**
   - Enter a country code, for example, **US**. Only two characters are permitted.
5. Indicate the number of days that the certificate is valid or accept the default of 365 days. For a CA certificate, you probably want to choose a longer period, for example, **1500** days.
6. Press **Enter** to continue, after the gskkyman utility successfully creates your certificate.
7. Indicate that the certificate key is the default key in the database by doing the following:
   - Enter **1** to mange your keys and certificates.

- Select **CA certificate for server1**.
- Enter **3** to set the key as the default.

Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.

8. Press **Enter** to continue working with the CA key database, after your request completes successfully.
9. Verify that your self-signed CA certificate was created:
   a. Enter **1** to show certificate information. The subject name and the issuer name should match.
   b. Enter **0** to return to the menu.
   c. Enter **2** to show key information. The default key should indicate **Yes**.
10. Press **Enter** to continue working with the CA key database.
11. Indicate that you want to export the CA certificate to a file by entering **6**.

    Note: You use this file in two places. You download it to your browser in "Step 5. Add your CA name to the list of trusted CAs for your browser" on page 89. You import it into your server key database in "Step 2. Set up the server key database"
12. Indicate that you want to save the output certificate file as Base64 ASN.1 DER by entering **2**.
13. Enter the export file named `cert.arm`. For more information on certificate file types, see "Certificate file types generated by the gskkyman utility" on page 157.
14. Press **Enter** to continue, then enter **0** to exit.

Step 2. Set up the server key database

A server key database is required for each server that supports secure SSL connections. You must specify the fully qualified path and name of the server key database on the KeyFile directive in the server configuration file, `httpd.conf`. You do not need to include the path and name of the CA key database on the KeyFile directive because it is not used by the Web server.

In this step, you create the following files:

- `server1.kdb` is the server key database used for SSL operations.
- `server1.rdb` is the server request database which contains the server public-private key pair. When the gskkyman utility generates this file, it uses the same name you specify for the server key database.
- `server1.sth` is the stash file containing the encrypted password for the server key database. This file is required for secure SSL connections and must be in the same directory as the server key database. If the Web server cannot find the stash file, the server starts with a non-secured connection. When the gskkyman utility generates this file, the utility uses the same name you specify for the server key database.
- `certreq.arm` is an ASCII file which contains your server public-private key pair and unsigned certificate. This is the default name.

Creating your server key database: To create your server key database:

1. Change to the path of the new server key database location by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.

   Note: For added security of the stash file, set the permissions for the subdirectory to 700.
2. Enter `gskkyman` to start the utility.

3. Enter **1** to create a new server key database.
4. Enter a unique name for your server key database. In this example, `server1.kdb` is used.
5. Enter a password using the guidelines in "Key database password" on page 158.
6. Enter the password again for verification.
7. Specify the number of days until the password expires, or never allow the password to expire.
8. Enter the database record length, or accept the default record length of 2500. The gskkyman utility displays a message confirming that the database is created.
9. Press **Enter** to continue working with the server key database.
10. Enter **10** to store the encrypted database password in a stash file. A message displays confirming that the stash file is created. The name of the stash file is the same as the database name, for example, `server1.sth`.

    Note: You must store the encrypted password for your server key database in a stash file. This file is required for secure SSL connections and must be in the same directory as the server key database. If this file is not created, the server starts but the connection is not secure.
11. Press **Enter** to continue working with the server key database.

Adding your CA name to the list of trusted CAs on the server: Add the name of your newly created CA to the list of trusted certificate authorities that your server recognizes. To add the name of your CA, import your CA certificate file, `cert.arm`, into the server key database:

1. Enter **7** to import a certificate.
2. Enter the CA certificate file name, `cert.arm`.
3. Enter the CA label, **CA certificate for server1**.
4. Press **Enter** to return to the Key Management menu, after your request completes successfully.
5. Verify that your CA is now listed as a trusted CA on your server:
   a. Enter **2** to manage certificates.
   b. Find and select your certificate label in the list, for example, **CA certificate for server1**
   c. Enter **1** to show CA certificate information. **Trusted** should indicate **Yes**.
6. Enter **0**, then press **Enter** until you return to the Key Management menu.

Creating your server certificate request: In this step, you create your server certificate request which includes your public-private key pair and unsigned certificate. To create your server certificate request:

1. Enter **4** to create a new certificate request.
2. Select one of the certificate types along with its key size. A smaller key size impacts performance less because it requires fewer resources to encrypt and decrypt than a larger key size. However, you need to determine the appropriate key size for your security needs. For more information on encryption and supported key sizes, see "Encryption support for the HTTP Server" on page 64.
3. Enter the request file name, `certreq.arm`. For information on certificate file types, see "Certificate file types generated by the gskkyman utility" on page 157.

4. Enter a name or label to use for identifying the key and certificate in the database, for example, **Certificate for server1**. Avoid using the word `request` when specifying labels.
5. Complete the fields that generate the Distinguished Name (DN):
   - Enter the common name or Domain Name of the server, for example, **server1.raleigh.ibm.com**.
   - Optionally enter an organization unit, for example, **IBM ID z/OS** .
   - Enter an organization name, for example, **IBM ID Raleigh**.
   - Optionally enter a city or locality, for example, **Raleigh**.
   - Optionally enter a state or province, for example, **North Carolina**
   - Enter a country code, for example, **US**. Only two characters are premitted.

   A message displays confirming that the request has completed successfully. The message indicates that the gskkyman facility created the `server1.rdb` and `certreq.arm` files.

   Note: Do not attempt to edit or move the `server1.rdb` file. If this file is not present or is corrupted when you attempt to receive your signed server certificate into the server key database, you have to resubmit your certificate request.
6. Press **Enter** to continue, then enter **0** to exit the gskkyman utility.

Step 3. Sign the server certificate

In this step you create a file that contains your signed server certificate. Specify a file name of your choice. In this example, the file name is `servcert.txt`.

In "Step 1. Set up the CA key database" on page 85, you set up your CA environment which enables you to act as your own CA and sign certificates. A signed server certificate is required before clients can establish an SSL connection to your server. Because you are acting as your own CA, you sign the server certificate request you created in "Step 2. Set up the server key database" on page 109. If you were using an external commercial CA, such as VeriSign, you would send the server certificate request to that CA for signature.

To sign your server certificate using the file names in this example, enter the following command:

```
gskkyman -g -x 365 -cr certreq.arm -ct servcert.txt -k cakey.kdb
```

To issue this command, you must have access to the certificate request file, `certreq.arm`, and the CA key database, `cakey.kdb`. If these files are not in your current working directory, enter the fully qualified file names on the command. You are prompted for the CA key database password.

- `-x 365` specifies that this certificate is valid for 365 days. This number must be less than the number of days the CA certificate is valid, which is `1500` in this example. If you do not create the server and CA certificates on the same day, the effective number of days starts to decrement.
- `certreq.arm` is the certificate request file you created in "Step 2. Set up the server key database" on page 86.
- `servcert.txt` is the file created by this command. This file contains your signed server certificate. Specify a file name of your choice.
- `cakey.kdb` is the name of the CA key database you created in "Step 1. Set up the CA key database" on page 85.

Step 4. Receive the signed server certificate into the server key database

To receive your signed certificate into the server key database:

1. Change to the path where the server key database is located by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.
2. Enter `gskkyman` to start the utility.
3. Enter **2** to open your server key database.
4. Enter the database name, `server1.kdb`.
5. Enter the database password.
6. Enter **5** to receive your server certificate.
7. Enter the name of your signed server certificate. In this example, that name is `servcert.txt`.
8. Press **Enter** to continue, after the gskkyman utility successfully receives your certificate.
9. Indicate that the certificate key is the default key in the database by doing the following:
   - Enter **1** to manage your keys and certificates.
   - Select **Certificate for server1**.
   - Enter **3** to select the key as the default.

   Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.
10. Press **Enter** to continue.
11. Enter **2** to show key information. The default key should indicate **Yes**.
12. Press **Enter**, then enter **0** to exit the gskkyman utility.

Step 5. Add your CA name to the list of trusted CAs for your browser

You need to add your CA name to the list of CAs known by your browser. To do this, download the `cert.arm` file you created in "Step 1. Set up the CA key database" on page 85 to your browser.

Before downloading the `cert.arm` file:

- Copy the `cert.arm` file into your document root directory. The document root directory is on the `Pass /*` directive in your server configuration file; the default directory is:

  ```
  /usr/lpp/internet/server_root/pub
  ```

  Note: If you put the `cert.arm` file in another directory, you need to add a Pass directive for the directory to the server configuration file if one does not already exist. Alter the URL to download the CA certificate to reflect the directory.

Download the CA certificate that uses the names in this example:

- Start your Web server.
- Open a browser.
- Enter the following URL:

  ```
  http://server1.raleigh.ibm.com/cert.arm
  ```

  – `server1.raleigh.ibm.com` is the fully qualified host name of the server.
  – `cert.arm` is the CA certificate file.

Follow the browser prompts to install the CA certificate. Newer versions of the Netscape and Microsoft Internet Explorer browsers automatically start a wizard to help you install the certificate. If you use Microsoft Internet Explorer, you may need to open the file rather than saving it to disk to start the wizard. See the online help in your browser or browser documentation for additional information. Generally for Netscape, if you receive a window asking if you would like to accept the CA to certify network sites, electronic mail (e-mail) users, and software developers, do so.

Step 6. Register the server key database with the Web server

To register your server key database in this example with the Web server, add the following KeyFile directive to your server configuration file, `httpd.conf`:

```
KeyFile  /ibm/security/user1/server1.kdb
```

Step 7. Use server security defaults

In this example, the following security default settings are used:

SSLMode
: Support for SSL connections is set **on**.

SSLPort
: Port 443 is the default port used for SSL.

SSLCipherSpec
: The server uses an encryption level, supported by both the client and the server.

You can change the Web server security defaults, by editing the appropriate security directives in the configuration file.

Step 8. Verify that you can establish a secure connection with the server

To verify that you can establish a secure connection with your server, start your Web server. Then start a browser and enter the following URL:

```
https://your_server_name:SSL_port_number
```

The URL for this example is:

```
https://serva.raleigh.ibm.com:443
```

Step 9. Optionally, set up client authentication

For instructions, see “Example 4A for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 4 or later)”.

Example 4A for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 4 or later)

Note: The gskkyman utility changed in z/OS Version 1 Release 4. Use this example only if your system is at Version 1 Release 4 or a later release.

This example shows how to set up client authentication using client certificates. Before using this example, set up secure connections using either of the following examples:

- “Example 2A for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 4 or later)” on page 78
- “Example 3A for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 4 or later)” on page 84

Alternatives for client certificate protection: A variety of ways exists to set up protection for client certificates. This example shows how to map a client certificate to a RACF user ID. For alternatives, see "Creating protection setups for Secure Sockets Layer (SSL) client authentication" on page 168.

CA certificate notes: You must insure that the CA that signs your client certificates is marked with a status of TRUST and that it is stored in your server key database. During the SSL handshake the server tells the client which CAs it trusts based on the trusted CAs in the server key database. The browser then searches its client certificates for ones issued by these CAs and allows the user to choose from a list and determine which certificate to send to the server. This example assumes that you use the same CA to sign your client certificate that you used to sign your server certificate. This implies that the CA certificate is already in your server key database and marked with a status of TRUST.

Note: If you run multiple Web servers, make sure that you store the server, Certificate Authority (CA), and client key databases for a particular Web server in a separate set of files from key databases belonging to other Web servers. Otherwise, if one of the files becomes corrupted, you lose key databases for multiple Web servers. SSL does not initialize for any of the Web servers whose server key databases are stored in a corrupted file.

RACF notes:

- The user ID issuing the RACF commands must have the proper authority.
- Choose the environment in which you execute the RACF commands (TSO READY, ISPF option 6, and so forth). Implementing these commands varies from one environment to another. See your local RACF administrator for assistance, or review the appropriate books for your environment. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Related information: The Web server provides support for retrieving security information from a Lightweight Directory Access Protocol (LDAP) server. For more information on LDAP support, see your Chapter 15, "Retrieving Lightweight Directory Access Protocol information" on page 297.

Summary of tasks

In this example, you perform the following main tasks:

1. Update server configuration directives
2. Insure the CA certificate is in the client browser
3. Provide options for obtaining client certificates, if acting as your own CA
4. Map a client certificate to RACF
5. Verify that the client can access a protected page with the client certificate

Step 1. Update server configuration directives

SSLClientAuth directive: The default setting for the SSLClientAuth directive is off. Because the server uses client certificates to validate clients and checks for CA certificates in the local database, specify:

```
SSLClientAuth  local
```

UserId directive:

Note: UserId directive refers to the default access control user ID for the Web server. UserId subdirective refers to the UserId specified on a Protection directive or protection setup. A UserId setting on a protection setup overrides the user ID specified on the UserId directive.

The initial configuration file setting for the UserId directive is:

```
UserId  %%CLIENT%%
```

In this example, %%CLIENT%% is used for the default access control user ID; %%CERTIF%% is generally used only if your entire Web site is protected. In this example, information in a specific directory is protected using client certificates, not the entire Web site. Therefore, %%CERTIF%% is specified only on the UserId subdirective on the Protection directive.

For more information on the UserId directive , see "Userid - Specify the Default Access Control user ID" on page 457.

Protection and Protect directives: For this example, add the following Protection and Protect directives to the server configuration file:

```
Protection PROTECTED_INFO {
       ServerId        Security_Administration
       AuthType        Basic
       PasswdFile      %%SAF%%
       UserId          %%CERTIF%%
       Mask            anybody
}

Protect  /secret/*  PROTECTED_INFO
```

Notes:

1. **ServerId**: Specify a name of your choice. Make this name unique for each protection setup.
2. **AuthType**: The AuthType subdirective lets you limit access based on user names and passwords. With an AuthType of Basic, passwords are sent to the client as plain text; they are encoded but not encrypted.
3. **PasswdFile**: For client certificates, the value of this field must be `%%SAF%%`.
4. **UserId**: For client certificates, the value of this field must be `%%CERTIF%%`. By specifying `%%CERTIF%%` on the UserId subdirective of the Protection directive, you indicate to use client certificates to control access to information only in the directory specified on the Protect directive. If you specify `%%CERTIF%%` on the UserId directive, client certificates are used to control access to all information on the Web site.
5. **Mask**: For client certificates, the value of the field must be one of the following:
   - `@`
   - `anybody`
   - `anyone`
   - `anonymous`
   - `anybody@(*)`
   - `anyone@(*)`
   - `anonymous@(*)`

   By using one of these values, the server completes requests without prompting for a user ID and password, as long as the client certificate maps to a RACF

user ID. However, if the client certificate is not associated with a RACF user ID, the server asks for the user ID and password. The browser pop-up window for the user ID and password contains the words `System_Logon` instead of the value on the `ServerID` subdirective.

6. The Protect directive must specify the name of the resource or directory that you are protecting, along with a reference to the Protection directive that defines the type of protection provided. The Protect directive in this example references the PROTECTED_INFO Protection directive. The information in the `/secret/` directory is protected using client certificates. Note that password protection is still the default for accessing protected information; a client who does not have a client certificate is prompted for a RACF user ID and password.
7. For more information on Protection subdirectives, see "Protection subdirectives" on page 444.

Step 2. Insure the CA certificate is in the client browser

Browsers vary as to whether they require the CA certificate that signs the client certificate to be in the browser. This example assumes that you insure that your CA certificate is added to your browser if it is not already there.

If you use an external commercial CA whose CA certificate is already loaded into the browser, you can skip this step and go to "Step 4. Map a client certificate to Resource Access Control Facility" on page 96. If the external CA certificate is not in the browser, contact the CA to obtain the CA certificate. Follow the CA directions for loading the certificate into the browser.

If you act as your own CA and use the same CA certificate to sign client certificates that you used to sign your server certificate, then clients may have already loaded the CA certificate into their browsers in "Step 5. Add your CA name to the list of trusted CAs for your browser" on page 89. If the the clients have not, they can dowload your CA certificate by opening a browser and entering URL:

```
http://server1.raleigh.ibm.com/cert.arm
```

- `server1.raleigh.ibm.com` is the fully qualified host name of your server.
- `cert.arm` is the file containing your CA certificate.

Step 3. If acting as your own CA, provide options for obtaining client certificates

If you use an external commercial CA, such as VeriSign or Thawte, to obtain your client certificates, you can skip this step and go to "Step 4. Map a client certificate to Resource Access Control Facility" on page 96.

If you act as your own CA, you can use the following options for providing client certificates. The steps to implement client authentication vary depending on the application.

- Create the client certificates yourself using the gskkyman utility, RACDCERT, or a CA software product such as the RACF PKISERV application, and send the signed certificates to clients who connect to your server.
- Request that clients generate their own certificate requests and send them to you for signature. This method requires that clients have access to the gskkyman utility or a CA software product such as the RACF PKISERV application.

To find out more about the RACF PKISERV application, go to URL: `http://www.ibm.com/s390/products/racf/goodies.html`.

You can use one or more of these options. This example shows steps for the first option using the gskkyman utility.

Creating your client key database: To create your client key database:

1. Change to the path of the new client key database location by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.
2. Enter `gskkyman` to start the utility.
3. Enter **1** to create a new client key database.
4. Enter a unique name for your client key database name. In this example, `client.kdb` is used.
5. Enter a password using the guidelines in "Key database password" on page 158.
6. Enter the password again for verification.
7. Specify the number of days until the password expires, or never allow the password to expire.
8. Enter the database record length, or accept the default record length of 2500. The gskkyman utility displays a message confirming that the database is created.
9. Press **Enter** to continue working with the client key database.

Creating a client certificate request: To create a client certificate:

1. Enter **4** to create a new certificate request.
2. Select one of the certificate types along with its key size. A smaller key size impacts performance less because it requires fewer resources to encrypt and decrypt than a larger key size. However, you need to determine the appropriate key size for your security needs. For more information on encryption and supported key sizes, see "Encryption support for the HTTP Server" on page 64.
3. Enter a client certificate request name. In this example, the certificate name is `client1.arm`. For information on certificate file types, see "Certificate file types generated by the gskkyman utility" on page 157.
4. Enter a name or label to use for identifying the key and certificate in the database, for example, **Certificate for Jane Smith**. Avoid using the word `request` when specifying labels.
5. Complete the fields that generate the Distinguished Name (DN):
   - Enter the common name of the client, for example, **Jane Smith**.
   - Optionally enter an organization unit, for example, **IBM ID z/OS** .
   - Enter an organization name, for example, **IBM ID Raleigh**.
   - Optionally enter a city or locality, for example, **Raleigh**.
   - Optionally enter a state or province, for example, **North Carolina**
   - Enter a country code, for example, **US**. Only two characters are permitted.

   A message displays confirming that the request has completed successfully. This indicates that gskkyman created the **client1.arm** file.
6. Press **Enter** to continue, then enter **0** to exit the gskkyman utility.

Signing a client certificate: To sign the client certificate request using the file names in this example, enter the following command:

```
gskkyman -g -x 365 -cr client1.arm -ct client1.txt -k cakey.kdb
```

To issue this command, you must have access to the certificate request file `client1.arm` and the CA key database `cakey.kdb`. If these files are not in your

current working directory, enter their fully qualified file names on the command. You are prompted for the CA key database password.

- `-x 365` specifies that this certificate is valid for 365 days. This number must be less than the number of days the CA certificate is valid, which is `1500` in this example. If the client and CA certificates are not created on the same day, the effective number of days starts to decrement for the CA certificate.
- `client1.arm` is the certificate request file created by the client.
- `client1.txt` is the file created by this command. This file contains the signed client certificate. You specify a file name of your choice.
- `cakey.kdb` is the name of the CA key database you created in "Step 1. Set up the CA key database" on page 85.

Importing the CA certificate into the client key database: Store the CA certificate that signed your client certificate in the client key database before the database can receive the client certificate. To store the CA certificate:

1. Change to the path where the client key database is located by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.
2. Enter **gskkyman** to start the utility.
3. Enter **2** to open your client key database.
4. Enter the database name, `client.kdb`.
5. Enter the database password.
6. Enter **7** to import the CA certificate.
7. Enter the certificate file name. In this example, the file was created in "Creating your self-signed CA certificate" on page 85. The file name is `cert.arm`.
8. Enter the label for the certificate, **CA certificate for server1**.
9. Press **Enter** to continue working with the client key database.

Receiving the signed client certificate into your client key database: To receive the signed client certificate into your client key database:

1. Enter **5** to receive the client certificate.
2. Enter the certificate file name. In this example, the file name is `client1.txt`.
3. Do not designate the certificate as the default key in the client key database.
4. Press **Enter** to continue, after the gskkyman utility successfully receives your certificate.

Sending the signed client certificate to the client:

Exporting the signed client certificate into a PKCS12 file: Export the signed client certificate into a PKCS12 file before loading it into your browser:

1. Enter **1** to manage keys and certificates.
2. Select the client certificate label, **Certificate for Jane Smith**.
3. Enter **7** to export the client certificate and client key to a file.
4. Specify the export file format by selecting one of the binary options. The gskkyman utility allows you to export the certificate in PKCS12 format with either binary or Base64 encoded ASCII. However, at least some of the major browsers do not accept the PKCS12 format in Base64 encoded ASCII.
5. Enter the export file name, for example, `client1.p12`.
6. Enter a password using the guidelines in "Key database password" on page 158.
7. Enter the password again for verification.
8. Select whether you want strong encryption, or the weaker export encryption.
9. Press **Enter**, then enter **0** to exit the gskkyman utility, after your request completes successfully.

Transferring the PKCS12 file to the client workstation: This example shows how to use the **FTP** command to transfer the PKCS12 file containing the signed client certificate to the client workstation. The following steps are performed on the workstation:

1. Enter the **FTP** command and the host name or IP address of the server, for example, **ftp server1.raleigh.ibm.com**.
2. Enter your user ID and password, when prompted.
3. Change to the directory where the client certificate PKCS12 file `client1.p12` is located, for example, **cd /ibm/security/user1**.
4. Enter **bin** to transfer the file in binary format.
5. Transfer the file to the workstation by entering **get client1.p12**.
6. Type **quit** to exit.

Loading the PKCS12 file into the client browser: This example shows how to load the PKCS12 file into the Netscape Communicator browser:

1. Start the browser.
2. **Click Communicator** → **Tools** → **Security Info**, to access the security information.
3. **Click Yours**, under **Certificates**.
4. **Click Import a Certificate**. You may need to scroll down to see this option.
5. Highlight the PKCS12 file.
6. **Click Open**, and enter the case sensitive password protecting the file.
7. **Click OK**. The following message displays: `Your certificates have been successfully imported.`
8. **Click OK**. You should see the certificate label in the window called **These are your certificates**. You may need to scroll down to find the label.

   Note: On browser versions prior to Netscape 4.6.1, there can be a problem displaying the label, for example, the label name may appear as **????@????**.

Step 4. Map a client certificate to Resource Access Control Facility

You can use the following Resource Access Control Facility (RACF) options to map a client certificate to RACF when the client certificate is created with some method other than RACF commands or a RACF application such as PKISERV:

- **Certificate Name Filtering function:** This function is available for OS/390 Release 10 and later.
- **Automatic registration of digital certificates on the Web:** The Autoregistration Web Application enables a client to automatically register a certificate with the Web server.
- **Using ISPF panels or the RACDCERT command:** These two options take the same inputs, but you use panels with ISPF and a command line with RACDCERT.

For information on these options, see the *z/OS Security Server (RACF): Security Administrator's Guide*. To access this guide and other RACF books on the Web, go to URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

You can use one or more of these options. This example shows you how to use the RACDCERT command.

In this example you:

- Store the client certificate in an MVS dataset.
- Associate the client certificate with a RACF user ID.

Export the client certificate into a file: RACF maps client certificates in PKCS12 format. Since some browsers require client certificates in PKCS12 format, we map a PKCS12 formatted file to a RACF user ID. If you act as your own CA, we use the same PKCS12 file sent to the client in "Sending the signed client certificate to the client" on page 95 (**client1.p12**) to map the client certificate to RACF. If you use an external commercial CA, you can export the client certificate from the browser. The PKCS12 file contains the private key and certificate.

Note: The client certificate can be output from the gskkyman utility in binary or base 64 encoded ASCII format for input to RACF. Some browsers can also output the client certificate in these formats or other formats for input to RACF. If either the binary or base 64 encoded ASCII format is output, the resulting file contains the certificate without the private key. The format of the client certificate in RACF can differ from the format of the certificate in the browser.

Placing the client certificate file in an MVS data set: You must store client certificates on MVS in variable block (VB) format. The client certificates in this example are in an HFS directory. Use the TSO/E **OGET** command to move the certificate file from the HFS directory into an MVS sequential data set. If you move the certificate into a new data set, the **OGET** command creates a VB sequential data set by default. You must use the **OGET** command to move the client certificate into the MVS sequential data set; exporting it from the browser to FTP it directly into the MVS data set does not work because the certificate file is not in the correct format.

Example:

```
oget '/ibm/security/user1/client1.p12' 'jsmith.client1.p12' binary
```

Note: You can execute this TSO/E command from TSO/E, ISPF option 6, and the shell. To find out more about this command, including where to execute it, please see the *z/OS UNIX System Services Command Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Installing the client certificate in RACF: To perform the following steps, ensure that you use an MVS user ID that has the authority to map a client certificate to an MVS user ID.

1. Issue the **RACDCERT** command to add the client certificate to the RACF data base for each client. For example:

   ```
   RACDCERT ID(RACFID1) ADD('JSMITH.CLIENT1.P12')
   WITHLABEL('Certificate for Jane Smith') TRUST PASSWORD('X2RL')
   ```

   - **RACFID1** is the RACF user ID under which the client certificate is added.
   - `'SERVER1.CLIENT1.P12'` is the name of the dataset where the certificate file is located.
   - **TRUST** indicates that you can use the client certificate to authenticate the user ID **RACFID1**.
   - **Certificate for Jane Smith** is the label of the client certificate.
   - **X2RL** is the required password for PKCS12 certificates.
2. Issue the following **SETROPTS** command to refresh the DIGTCERT class for all the clients:

   ```
   SETROPTS RACLIST(DIGTCERT) REFRESH
   ```

3. Verify that the client certificate is associated with a user ID that has been defined to RACF, by issuing the following **RACDCERT** command and specifying the user ID in the ID field:

   ```
   RACDCERT ID(RACFID1) LIST
   ```

   If you are logged onto the user ID you are trying to verify, you can issue the RACDCERT command without operands to display the certificate for the user ID.

Step 5. Verify that the client can access a protected page with the client certificate

To test that client authentication is set up correctly for the client in this example:

1. Start your Web server.
2. Start the browser and specify URL:

   ```
   https://server1.raleigh.ibm.com/secret/budget.html
   ```

3. Select the label for the client certificate, when prompted by the browser. If the setup is correct, you view the page without prompts for a user ID and password.

Example 1 for gskkyman: Setting up secure connections using a self-signed server certificate (z/OS Version 1 Release 3 or earlier)

This example is a quick start method for setting up SSL on your server.

Note: The gskkyman utility changed in z/OS Version 1 Release 4. Use this example only if your system is at Version 1 Release 3 or an earlier release.

Note: Using a self-signed server certificate is recommended only for test environments. This method gives you a quick way to test your SSL setup and internal applications with a small number of trusted clients.

Note: If you run multiple Web servers, make sure that you store the server, Certificate Authority (CA), and client key databases for a particular Web server in a separate set of files from key databases belonging to other Web servers. Otherwise, if one of the files becomes corrupted, you lose key databases for multiple Web servers. SSL does not initialize for any of the Web servers whose server key databases are stored in a corrupted file.

System SSL notes:

You must use System SSL to establish secure connections. To use System SSL with the Web server, the System SSL load library must exist in linklist and must be under program control. If you have not already done so:

- Add the load library to the linklist.
- Turn on program control for the library by issuing the following RACF commands from a user ID that has the proper authority:

  ```
  RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
  SETROPTS WHEN(PROGRAM) REFRESH
  ```

  If turning on program control for the first time, use the RDEFINE command instead of the RALTER command.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Summary of tasks

In this example, you will perform the following main tasks:

1. Create the server key database
2. Create your self-signed server certificate
3. Register the server key database with the Web server
4. Use server security defaults
5. Verify that you can establish a secure connection with the server

Step 1. Create the server key database

A server key database is required for each server that supports secure SSL connections. You must specify the fully qualified path and name of the server key database on the KeyFile directive in the server configuration file, `httpd.conf`.

In this step, you will create the following files:

- `serva.kdb` is the server key database used for SSL operations.
- `serva.rdb` is the server request database which contains the server public-private key pair. When the gskkyman utility generates this file, it uses the same name you specify for the server key database.
- `serva.sth` is the stash file containing the encrypted password for the server key database. This file is required for secure SSL connections and must be in the same directory as the server key database. If the Web server cannot find the stash file, the server starts with a non-secured connection. When gskkyman generates this file, the utility uses the same name you specify for the server key database.

To create your server key database:

1. Change to the path of the new server key database location by entering `cd` *`path`*. In this example, *path* is `/u/serva`.

   Note: For added security of the stash file, set the permissions for the subdirectory to 700.
2. Enter `gskkyman` to start the utility.
3. Enter **1** to create a new server key database.
4. Enter a unique name for your server key database. In this example, `serva.kdb` is used.
5. Enter a password using the guidelines in "Key database password" on page 158.
6. Enter the password again for verification.
7. Indicate whether the password should expire. If yes, accept the default of 60 days or enter the number of days. A message is displayed confirming that the database has been created.
8. Enter **1** to continue working with the server key database.
9. Enter **11** to store the encrypted database password in a stash file. A message is displayed confirming that the stash file has been created. The name of the stash file is the same as the database name, for example, **serva.sth**.

   Note: You must store the encrypted password for your server key database in a stash file. This file is required for secure SSL connections and must be

in the same directory as the server key database. If this file is not created, the server will start but the connection will not be secure.

10. Enter **0** to continue working with the server key database.

Step 2. Create your self-signed server certificate

In this step, you will create the following file:

- `cert.arm` is an ASCII file which contains your server public-private key pair and self-signed server certificate. This is the default name.

To create your self-signed server certificate:

1. Enter **5** to create a self-signed server certificate.
2. Enter the version number. The higher the version number, the more fields are included in the certificate. For example, a version **3** certificate request includes all possible fields.
3. Enter a name (label) that is used to identify the key and certificate in the database, for example, **Certificate for serva**.
4. Enter the number of the key size you want to use. A smaller key size impacts performance less because it requires fewer resources to encrypt and decrypt than a larger key size. However, you need to determine the appropriate key size for your security needs. For more information on encryption and supported key sizes, see "Encryption support for the HTTP Server" on page 64.
5. Complete the fields that generate the Distinguished Name (DN):
   - Enter the common name (Domain Name) of the server, for example, **serva.raleigh.ibm.com**.
   - Enter an organization name, for example, **IBM ID Raleigh**.
   - Optionally enter an organization unit, for example, **IBM ID z/OS**.
   - Optionally enter a city or locality, for example, **Raleigh**.
   - Optionally enter a state or province, for example, **North Carolina**
   - Enter a country code, for example, **US**. Only 2 characters are allowed.
6. Indicate the number of days that the certificate will be valid or accept the default of 365 days.
7. Indicate that the certificate will be the default key in the database.

   Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.
8. Indicate that you want to save the certificate to a file by entering **1**.
9. Indicate that you want to save the file as Base64 encoded ASCII by entering **1**.
10. Press **Enter** to use the default certificate name, `cert.arm`. For more information on certificate file types, see "Certificate file types generated by the gskkyman utility" on page 157.
11. After your request has completed successfully, enter **0** to continue working with the server key database.
12. To verify that your self-signed server certificate was created:
    a. Enter **1** to list your keys and certificates.
    b. Enter **1** to view the new self-signed server certificate, **Certificate for serva**.
    c. Enter **1** to show key and certificate information. The issuer and subject name should be the same.
    d. To exit, continue pressing **Enter** until you return to the Key database menu, then enter **0** to exit gskkyman.

Step 3. Register the server key database with the Web server

To register the server key database in this example with the Web server, add the following KeyFile directive to your server configuration file, httpd.conf:

```
KeyFile  /u/serva/serva.kdb
```

Step 4. Use server security defaults

In this example, the following security default settings are used:

SSLMode
Support for SSL connections is set **on**.

SSLPort
Port 443 is the default port used for SSL.

SSLCipherSpec
The server uses an encryption level, supported by both the client and the server.

Step 5. Verify that you can establish a secure connection with the server

To verify that you can establish a secure connection with your server, start your Web server. Then start a browser and enter the following URL:

```
https://your_server_name:SSL_port_number
```

The URL for this example is:

```
https://serva.raleigh.ibm.com:443
```

Example 2 for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 3 or earlier)

This example includes steps for using an external commercial CA to sign your server certificate.

Note: The gskkyman utility changed in z/OS Version 1 Release 4. Use this example only if your system is at Version 1 Release 3 or an earlier release.

Note: Using an external commercial CA, such as VeriSign or Thawte, is recommended for Internet connections and e-business. This method provides the highest level of security. You can optionally use client authentication to verify the identity of those accessing your server.

Note: If you run multiple Web servers, make sure that you store the server, Certificate Authority (CA), and client key databases for a particular Web server in a separate set of files from key databases belonging to other Web servers. Otherwise, if one of the files becomes corrupted, you lose key databases for multiple Web servers. SSL does not initialize for any of the Web servers whose server key databases are stored in a corrupted file.

System SSL notes:
You must use System SSL to establish secure connections. To use System SSL with the Web server, the System SSL load library must exist in linklist and must be under program control. If you have not already done so:

- Add the load library to the linklist.
- Turn on program control for the library by issuing the following RACF commands from a user ID that has the proper authority:

  ```
  RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
  SETROPTS WHEN(PROGRAM) REFRESH
  ```

  If turning on program control for the first time, use the RDEFINE command instead of the RALTER command.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Summary of tasks

In this example, you will perform the following main tasks:

1. Set up the server key database
2. Send your server certificate request to an external commercial CA
3. If necessary, add the CA to the list of trusted CAs on the server
4. Receive your signed server certificate into the server key database
5. Use server security defaults
6. Verify that you can establish a secure connection with the server
7. Optionally, set up client authentication using client certificates

Step 1. Set up the server key database

A server key database is required for each server that supports secure SSL connections. You must specify the fully qualified path and name of the server key database on the KeyFile directive in the server configuration file, `httpd.conf`.

In this step, you will create the following files:

- `server1.kdb` is the server key database used for SSL operations.
- `server1.rdb` is the server request database which contains the server public-private key pair. When the gskkyman utility generates this file, it uses the same name you specify for the server key database.
- `server1.sth` is the stash file containing the encrypted password for the server key database. This file is required for secure SSL connections and must be in the same directory as the server key database. If the Web server cannot find the stash file, the server starts with a non-secured connection. When the gskkyman utility generates this file, the utility uses the same name you specify for the server key database.
- `certreq.arm` is an ASCII file which contains your server public-private key pair and unsigned certificate. This is the default name.

Creating your server key database: To create your server key database:

1. Change to the path of the new server key database location by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.

   Note: For added security of the stash file, set the permissions for the subdirectory to 700.
2. Enter `gskkyman` to start the utility.
3. Enter **1** to create a new server key database.
4. Enter a unique name for your server key database. In this example, `server1.kdb` is used.
5. Enter a password using the guidelines in "Key database password" on page 158.
6. Enter the password again for verification.
7. Indicate whether the password should expire. If yes, accept the default of 60 days or enter the number of days. A message is displayed confirming that the database has been created.
8. Enter **1** to continue working with the server key database.
9. Enter **11** to store the encrypted database password in a stash file. A message is displayed confirming that the stash file has been created. The name of the stash file is the same as the database name, for example, `server1.sth`.

Note: You must store the encrypted password for your server key database in a stash file. This file is required for secure SSL connections and must be in the same directory as the server key database. If the Web server cannot find the stash file, the server will start with a non-secured connection.

10. Enter **0** to continue working with the server key database.

Creating your server certificate request: In this step, you will create your server certificate request which includes your public-private key pair and unsigned certificate. To create your server certificate request:

1. Enter **3** to create a new key pair and certificate request.
2. Press **Enter** to use the default filename, `certreq.arm`. The certificate request file is in Base 64 encoded ASCII format. For more information on certificate file types, see "Certificate file types generated by the gskkyman utility" on page 157.
3. Enter a name or label to use for identifying the key and certificate in the database, for example, **Certificate for server1**. Avoid using the word `request` when specifying labels.
4. Enter the number of the key size you want to use. A smaller key size impacts performance less because it requires fewer resources to encrypt and decrypt than a larger key size. However, you need to determine the appropriate key size for your security needs. For more information on encryption and supported key sizes, see "Encryption support for the HTTP Server" on page 64.
5. Complete the fields that generate the Distinguished Name (DN):
   - Enter the common name (Domain Name) of the server, for example, **server1.raleigh.ibm.com**.
   - Enter an organization name, for example, **IBM ID Raleigh**.
   - Optionally enter an organization unit, for example, **IBM ID z/OS** .
   - Optionally enter a city or locality, for example, **Raleigh**.
   - Optionally enter a state or province, for example, **North Carolina**
   - Enter a country code, for example, **US**. Only 2 characters are allowed.

   A message is displayed confirming that the request has completed successfully. This indicates that gskkyman created the `server1.rdb` and `certreq.arm` files.

   Note: Do not attempt to edit or move the `server1.rdb` file. If this file is not present or is corrupted when you attempt to receive your signed server certificate into the server key database, you will have to resubmit your certificate request.
6. Enter **1** to exit the gskkyman utility.

Step 2. Send your server certificate request to an external commercial certificate authority

After you create your server certificate request, you send that request to a certificate authority (CA) to be signed. First transfer the file containing the server certificate request to your workstation. Then follow the CAs instructions for sending and receiving your certificate request.

By default, the following CAs are designated as trusted on the Web server:

VeriSign

For more information on the following certificates, go to the VeriSign Web site at URL `http://www.verisign.com/`.

- VeriSign Test CA Root Certificate

- RSA Secure Server Certification Authority
- VeriSign Class 1 Public Primary Certification Authority
- VeriSign Class 2 Public Primary Certification Authority
- VeriSign Class 3 Public Primary Certification Authority
- VeriSign Class 4 Public Primary Certification Authority

Thawte

For more information on the following certificates, go to the Thawte Web site at URL `http://www.thawte.com/`.

- Thawte Personal Freemail CA
- Thawte Personal Basic CA
- Thawte Premium Server CA
- Thawte Server CA

Transferring the server certificate request to your workstation: This example shows how to use the File Transfer Protocol (FTP) command to transfer the `certreq.arm` file containing the server certificate request to your workstation. Perform the following steps on the workstation:

1. Enter the **FTP** command and either the host name or the IP address of the server. For example, enter `ftp server1.raleigh.ibm.com`.
2. Enter your user ID and password when prompted.
3. Change to the directory where you put the file containing the server certificate request, `certreq.arm`. For example, enter `cd /ibm/security/user1`.
4. Transfer the file to the workstation in ASCII format by entering `get certreq.arm`.
5. Type `quit` to exit.

Step 3. If necessary, add the CA to the list of trusted CAs on the server

If you send your certificate request to a CA that is already a trusted CA on your server, you can skip this step. In this step you first check if the CA is trusted on your server. If it is not, contact the CA to obtain their CA certificate. Transfer the CA certificate from your workstation to the host. Then import the CA certificate into your server key database and mark it as **Trusted**.

Checking the list of trusted CAs: To check the current list of trusted CAs on your server:

1. Change to the path where the server key database is located by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.
2. Enter **gskkyman** to start the utility.
3. Enter **2** to open the server key database.
4. Enter the server key database name, `server1.kdb`.
5. Enter the database password.
6. Enter **10** to list all trusted CAs. If the CA is not trusted, you must store the CA certificate in your server key database before you can receive your signed server certificate.
7. Enter **0** to exit gskkyman.

Transferring the CA certificate from your workstation to the host: This example shows how to use the File Transfer Protocol (FTP) command to transfer the file containing your commercial CA certificate to the host. In this example the file name is `cert.arm`. Perform the following steps on the workstation:

1. Enter the **FTP** command and either the host name or the IP address of the server, for example, **ftp server1.raleigh.ibm.com**.
2. Enter your user ID and password, when prompted.

3. Change to the directory where you are putting the file containing the CA certificate, `cert.arm`, for example, **cd /ibm/security/user1**.
4. Transfer the file to the host in ASCII format by entering **put cert.arm**.
5. Type **quit** to exit.

Storing the CA certificate in your server key database: To designate a CA as trusted on your server, store the certificate in your server key database:

1. Change to the path where the server key database is located by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.
2. Enter **gskkyman** to start the utility.
3. Enter **2** to open the server key database.
4. Enter your server key database name, `server1.kdb`.
5. Enter the database password.
6. Enter **6** to store a CA certificate.
7. Enter the name of the certificate.
8. Enter a label for the certificate.
9. When your request has completed successfully, enter **1** to return to the key database menu.
10. To verify that the CA is now listed as a trusted CA on your server, enter **10** to list all trusted CAs.
11. Enter **0** to exit gskkyman.

Step 4. Receive your signed server certificate into the server key database

In this step, you will create the following file:

- `servcert.arm` is an ASCII file which contains the server public-private key pair and certificate signed by the external commercial CA.

Example of a signed server certificate: The certificate you receive should contain `BEGIN CERTIFICATE` and `END CERTIFICATE` lines with data between as shown in the following example. If your certificate contains additional information before `BEGIN CERTIFICATE` or after `END CERTIFICATE`, do not include the extraneous information in the certificate file you create in this step.


```
-----BEGIN CERTIFICATE-----
 MIIB0DCCATkCBDV8PgswDQYJKoZIhvcNAQECBQAwcjELMAkGA1UEBhMCVVMxDTAL
 BgNVBAgTBE4uQy4xDDAKBgNVBAcTA1JUUDEMMAoGA1UEChMDSUJNMRcwFQYDVQQL
 Ew5XZWJzZXJ2ZXIgVGVzdDEfMB0GA1UEAxMWbXZzMTY3LnJhbGVpZ2guaWJtLmNv
 bTAaFws5ODA2MDgxOTM5WhcLOTkwNjA4MTkzOVowNDELMAkGA1UEBhMCVVMxDDAK
 BgNVBAoTA0lCTTEXMBUGA1UEAxMOcGtjczEwLmlibS5jb20wXDANBgkqhkiG9w0B
 AQEFAANLADBIAkEA1IYGldVmnKAI8hJQGT074oXTD0Tb+jFN8wkPqc+DVhYix1fj
 h/sbiuDZF66BMh5hnHfJr75633CgjW10EpID0wIDAQABMA0GCSqGSIb3DQEBAgUA
 A4GBAF1KVppAM7Gh2F9BBIY/jPMFlRp8+HAAVkK29Q4DxeF2FrTzQutKmO8duCWv
 xnJo4pgl5Uj29DSAsrX8mULfczyuZwVVXiCGnhN03pYj8bbQjo0edqQ7hYsRl3P4
 C72I+yRwtWUukfVgwdo0mWXyEclx7eT5jsW4weVEqWvuht8j
-----END CERTIFICATE-----
```


Before you can receive the signed certificate into your server key database, create a certificate file and copy your signed server certificate into that file. This example assumes that you create the certificate file on your workstation and that you name it `servcert.arm`. Include the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines and all data in between as shown in the example. Remove any extraneous information that the CA included in the file.

Transferring the signed server certificate from your workstation to the host: This example shows how to use the File Transfer Protocol (FTP) command to transfer

the file containing the signed server certificate to the host. In this example the file name is `servcert.arm`. Perform the following steps on the workstation:

1. Enter the **FTP** command and either the host name or the IP address of the server. For example, enter `ftp server1.raleigh.ibm.com`.
2. Enter your user ID and password when prompted.
3. Change to the directory where you are putting the file containing the signed server certificate, `servcert.arm`. For example, enter `cd /ibm/security/user1`.
4. Transfer the file to the host in ASCII format by entering `put servcert.arm`.
5. Type **quit** to exit.

To receive your signed certificate into the server key database:

1. Change to the path where the server key database is located by entering **cd** *path*. In this example, *path* is **/ibm/security/user1**.
2. Enter **gskkyman** to start the utility.
3. Enter **2** to open your server key database.
4. Enter the database name, `server1.kdb`.
5. Enter the database password.
6. Enter **4** to receive your server certificate.
7. Enter the certificate file name. In this example, the file name is `servcert.arm`.
8. Indicate that you want the certificate to be the default key in the key database.

   Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.
9. Enter **1** to exit gskkyman.

Step 5. Register the server key database with the server

To register your server key database in this example with the Web server, add the following KeyFile directive to your server configuration file, `httpd.conf`:

```
KeyFile  /ibm/security/user1/server1.kdb
```

Step 6. Use server security defaults

In this example, the following security default settings are used:

SSLMode
: Support for SSL connections is set **on**.

SSLPort
: Port 443 is the default port used for SSL.

SSLCipherSpec
: The server uses an encryption level, supported by both the client and the server.

You can change the Web server security defaults, by editing the appropriate security directives in the configuration file.

Step 7. Verify that you can establish a secure connection with the server

To verify that you can establish a secure connection with your server, start your Web server. Then start a browser and enter the following URL:

```
https://your_server_name:SSL_port_number
```

The URL for this example is:

```
https://serva.raleigh.ibm.com:443
```

Step 8. Optionally, set up client authentication using client certificates

For instructions, see "Example 4 for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 3 or earlier)" on page 113.

Example 3 for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 3 or earlier)

This example includes steps for setting up and administering your own certificate authority.

Note: The gskkyman utility changed in z/OS Version 1 Release 4. Use this example only if your system is at Version 1 Release 3 or an earlier release.

Note: Acting as your own CA is recommended only for test environments and private intranets. With this method, you set up your own CA and sign certificates. You can optionally use client authentication to verify the identity of those accessing your server.

Note: If you run multiple Web servers, make sure that you store the server, Certificate Authority (CA), and client key databases for a particular Web server in a separate set of files from key databases belonging to other Web servers. Otherwise, if one of the files becomes corrupted, you lose key databases for multiple Web servers. SSL does not initialize for any of the Web servers whose server key databases are stored in a corrupted file.

System SSL notes:

You must use System SSL to establish secure connections. To use System SSL with the Web server, the System SSL load library must exist in linklist and must be under program control. If you have not already done so:

- Add the load library to the linklist.
- Turn on program control for the library by issuing the following RACF commands from a user ID that has the proper authority:

```
RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

  If turning on program control for the first time, use the RDEFINE command instead of the RALTER command.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Summary of tasks

In this example, you will perform the following main tasks:

1. Set up the CA key database
2. Set up the server key database
3. Sign your server certificate
4. Receive your signed server certificate into the server key database
5. Add your CA name to the list of trusted CAs for your browser
6. Register the server key database with the Web server
7. Use server security defaults
8. Verify that you can establish a secure connection with the server
9. Optionally, set up client authentication using client certificates

Step 1. Set up the CA key database

In this step, you will create the following files:

- `cakey.kdb` is the CA key database.
- `cert.arm` is an ASCII file which contains your server public-private key pair and self-signed CA certificate. This is the default name.

Creating your CA key database:

1. Change to the path of the new CA key database location by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.
2. Enter `gskkyman` to start the utility.
3. Enter **1** to create a new CA key database.
4. Enter a unique name for your CA key database name. In this example, `cakey.kdb` is used.
5. Enter a password using the guidelines in "Key database password" on page 158.
6. Enter the password again for verification.
7. Indicate whether the password should expire. If yes, accept the default of 60 days or enter the number of days. A message is displayed confirming that the database has been created.
8. Enter **1** to continue working with the CA key database.

Creating your self-signed CA certificate:

1. Enter **5** to create a self-signed CA certificate.
2. Enter the version number. The higher the version number, the more fields are included in the certificate. For example, a version **3** certificate request includes all possible fields.
3. Enter a name (label) that is used to identify the key and certificate in the database, for example, **CA certificate for server1**.
4. Enter the number of the key size you want to use. A smaller key size impacts performance less because it requires fewer resources to encrypt and decrypt than a larger key size. However, you need to determine the appropriate key size for your security needs. For more information on encryption and supported key sizes, see "Encryption support for the HTTP Server" on page 64.
5. Complete the fields that generate the Distinguished Name (DN):
   - Enter a common name for your CA, for example, **IBM Raleigh z/OS ID server CA**.
   - Enter an organization name, for example, **IBM ID Raleigh**.
   - Optionally enter an organization unit, for example, **IBM ID z/OS** .
   - Optionally enter a city or locality, for example, **Raleigh**.
   - Optionally enter a state or province, for example, **North Carolina**
   - Enter a country code, for example, **US**. Only 2 characters are allowed.
6. Indicate the number of days that the certificate will be valid or accept the default of 365 days. For a CA certificate, you will probably want to choose a longer period, for example, **1500** days.
7. Indicate that the certificate will be the default key in the database. To sign server and client certificate, your CA certificate must be the default key in the database.

   Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.

8. Indicate that you want to save the certificate to a file by entering **1**.

   Note: This file will be downloaded to your browser in "Step 5. Add your CA name to the list of trusted CAs for your browser" on page 112.
9. Indicate that you want to save the output certificate file in Base64 encoded ASCII format by entering **1**.
10. Press **Enter** to use the default certificate name `cert.arm`. For more information on certificate file types, see "Certificate file types generated by the gskkyman utility" on page 157.
11. Enter **0** to return to the Key database menu.
12. To verify that your self-signed CA certificate was created:
    a. Enter **1** to list your keys and certificates.
    b. Enter **1** to view the new self-signed server certificate, `CA certificate for server1`.
    c. Enter **1** to show key and certificate information.

Step 2. Set up the server key database

A server key database is required for each server that supports secure SSL connections. You must specify the fully qualified path and name of the server key database on the KeyFile directive in the server configuration file, `httpd.conf`. You do not need to include the path and name of the CA key database on the KeyFile directive because it is not used by the Web server.

In this step, you will create the following files:

- `server1.kdb` is the server key database used for SSL operations.
- `server1.rdb` is the server request database which contains the server public-private key pair. When the gskkyman utility generates this file, it uses the same name you specify for the server key database.
- `server1.sth` is the stash file containing the encrypted password for the server key database. This file is required for secure SSL connections and must be in the same directory as the server key database. If the Web server cannot find the stash file, the server starts with a non-secured connection. When the gskkyman utility generates this file, the utility uses the same name you specify for the server key database.
- `certreq.arm` is an ASCII file which contains your server public-private key pair and unsigned certificate. This is the default name.

Creating your server key database: To create your server key database:

1. Change to the path of the new server key database location by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.

   Note: For added security of the stash file, set the permissions for the subdirectory to 700.
2. Enter `gskkyman` to start the utility.
3. Enter **1** to create a new server key database.
4. Enter a unique name for your server key database. In this example, `server1.kdb` is used.
5. Enter a password using the guidelines in "Key database password" on page 158.
6. Enter the password again for verification.
7. Indicate whether the password should expire. If yes, accept the default of 60 days or enter the number of days. A message is displayed confirming that the database has been created.
8. Enter **1** to continue working with the server key database.

9. Enter **11** to store the encrypted database password in a stash file. A message is displayed confirming that the stash file has been created. The name of the stash file is the same as the database name, for example, `server1.sth`.

   Note: You must store the encrypted password for your server key database in a stash file. This file is required for secure SSL connections and must be in the same directory as the server key database. If the Web server cannot find the stash file, the server will start with a non-secured connection.

10. Enter **0** to continue working with the server key database.

Adding your CA name to the list of trusted CAs on the server: You must add the name of your newly created CA to the list of trusted certificate authorities that are recognized by your server. To add the name of your CA, you need to import your CA key from your CA key database into the server key database:

1. Enter **8** to import keys.
2. Enter **1** to import keys from another database.
3. When prompted, enter the CA key database name `cakey.kdb` and the CA key database password.
4. Enter the number of the key to import. In this example, the key is **IBM ID CA on rtp1**. A message is displayed when the CA key has been successfully imported. Once the CA key has been imported into the server key database, it is automatically marked as a trusted CA.
5. Enter **0** to return to the Key database menu.
6. To verify that your CA is now listed as a trusted CA on your server, enter **10** to list all trusted CAs. To return to the Key database menu, enter **0**.

Creating your server certificate request: In this step, you will create your server certificate request which includes your public-private key pair and unsigned certificate. To create your server certificate request:

1. Enter **3** to create a new key pair and certificate request.
2. Press **Enter** to use the default filename, `certreq.arm`. The certificate request file is in Base 64 encoded ASCII format. For more information on certificate file types, see "Certificate file types generated by the gskkyman utility" on page 157.
3. Enter a name or label to use for identifying the key and certificate in the database, for example, **Certificate for server1**. Avoid using the word `request` when specifying labels.
4. Enter the number of the key size you want to use. A smaller key size impacts performance less because it requires fewer resources to encrypt and decrypt than a larger key size. However, you need to determine the appropriate key size for your security needs. For more information on encryption and supported key sizes, see "Encryption support for the HTTP Server" on page 64.
5. Complete the fields that generate the Distinguished Name (DN):
   - Enter the common name (Domain Name) of the server, for example, **server1.raleigh.ibm.com**.
   - Enter an organization name, for example, **IBM ID Raleigh**.
   - Optionally enter an organization unit, for example, **IBM ID z/OS** .
   - Optionally enter a city or locality, for example, **Raleigh**.
   - Optionally enter a state or province, for example, **North Carolina**
   - Enter a country code, for example, **US**. Only 2 characters are allowed.

   A message is displayed confirming that the request has completed successfully. This indicates that gskkyman created the `server1.rdb` and `certreq.arm` files.

Note: Do not attempt to edit or move the `server1.rdb` file. If this file is not present or is corrupted when you attempt to receive your signed server certificate into the server key database, you will have to resubmit your certificate request.

6. Enter **1** to exit the gskkyman utility.

Step 3. Sign the server certificate

In this step, you will create the following file:

- `servcert.txt` is the file which contains your signed server certificate. You specify a filename of your choice.

In "Step 1. Set up the CA key database" on page 108, you set up your CA environment which enables you to act as your own CA and sign certificates. A signed server certificate is required before clients can establish an SSL connection to your server. Because you are acting as your own CA, you will sign the server certificate request you created in "Step 2. Set up the server key database" on page 109. If you were using an external commercial CA, such as VeriSign, you would send the server certificate request to that CA for signature.

To sign your server certificate using the file names in this example, enter the following command:

```
gskkyman -g -x 365 -cr certreq.arm -ct servcert.txt -k cakey.kdb
```

To issue this command, you must have access to the certificate request file (`certreq.arm`) and the CA key database (`cakey.kdb`). If these files are not in your current working directory, enter the fully qualified filenames on the command. You will also be prompted for the CA key database password.

- `-x 365` specifies that this certificate is valid for 365 days. This number must be less than the number of days the CA certificate is valid, which is `1500` in this example. If you do not create the server and CA certificates on the same day, the effective number of days starts to decrement.
- `certreq.arm` is the certificate request file you created in "Step 2. Set up the server key database" on page 109.
- `servcert.txt` is the file created by this command. This file contains your signed server certificate. You specify a filename of your choice.
- `cakey.kdb` is the name of the CA key database you created in "Step 1. Set up the CA key database" on page 108.

Step 4. Receive the signed server certificate into the server key database

To receive your signed certificate into the server key database:

1. Change to the path where the server key database is located by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.
2. Enter **gskkyman** to start the utility.
3. Enter **2** to open your server key database.
4. Enter the database name, `server1.kdb`.
5. Enter the database password.
6. Enter **4** to receive your server certificate.
7. Enter the name of the certificate you created in "Step 3. Sign the server certificate". In this example, that name is `servcert.txt`.
8. Indicate that you want the certificate to be the default key in the key database.

Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.

9. Enter **1** to exit gskkyman.

Step 5. Add your CA name to the list of trusted CAs for your browser

You need to add your CA name to the list of CAs that are known by your browser. To do this, download the **cert.arm** file you created in "Step 1. Set up the CA key database" on page 108 to your browser.

Before downloading the `cert.arm` file:

- Copy the `cert.arm` file into your document root directory. The document root directory is on the `Pass /*` directive in your server configuration file; the default directory is:

  ```
  /usr/lpp/internet/server_root/pub
  ```

  Note: If you put the `cert.arm` file in another directory, you need to add a Pass directive for the directory to the server configuration file if one does not already exist. Alter the URL to download the CA certificate to reflect the directory.

Download the CA certificate that uses the names in this example:

- Start your Web server.
- Open a browser.
- Enter the following URL:

  ```
  http://server1.raleigh.ibm.com/cert.arm
  ```

  – `server1.raleigh.ibm.com` is the fully qualified host name of the server.
  – `cert.arm` is the CA certificate file.

Follow the browser prompts to install the CA certificate. Newer versions of the Netscape and Microsoft Internet Explorer browsers automatically start a wizard to help you install the certificate. If you use Microsoft Internet Explorer, you may need to open the file rather than saving it to disk to start the wizard. See the online help in your browser or browser documentation for additional information. Generally for Netscape, if you receive a window asking if you would like to accept the CA to certify network sites, electronic mail (e-mail) users, and software developers, do so.

Step 6. Register the server key database with the Web server

To register your server key database in this example with the Web server, add the following KeyFile directive to your server configuration file, `httpd.conf`:

```
KeyFile  /ibm/security/user1/server1.kdb
```

Step 7. Use server security defaults

In this example, the following security default settings are used:

SSLMode
Support for SSL connections is set **on**.

SSLPort
Port 443 is the default port used for SSL.

SSLCipherSpec
The server uses an encryption level, supported by both the client and the server.

You can change the Web server security defaults, by editing the appropriate security directives in the configuration file.

Step 8. Verify that you can establish a secure connection with the server

To verify that you can establish a secure connection with your server, start your Web server. Then start a browser and enter the following URL:

```
https://your_server_name:SSL_port_number
```

The URL for this example is:

```
https://serva.raleigh.ibm.com:443
```

Step 9. Optionally, set up client authentication

For instructions, see "Example 4 for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 3 or earlier)".

Example 4 for gskkyman: Setting up client authentication using client certificates (z/OS Version 1 Release 3 or earlier)

Note: The gskkyman utility changed in z/OS Version 1 Release 4. Use this example only if your system is at Version 1 Release 3 or an earlier release.

This example shows how to set up client authentication using client certificates. Before using this example, you must set up secure connections using either of the following examples:

- "Example 2 for gskkyman: Setting up secure connections using certificates signed by an external commercial CA (z/OS Version 1 Release 3 or earlier)" on page 101
- "Example 3 for gskkyman: Setting up secure connections using certificates signed by an internal CA (z/OS Version 1 Release 3 or earlier)" on page 107

Alternatives for client certificate protection: A variety of ways exists to set up protection for client certificates. This example shows how to map a client certificate to a RACF user ID. For alternatives, see "Creating protection setups for Secure Sockets Layer (SSL) client authentication" on page 168.

CA certificate notes: You must insure that the CA that signs your client certificates is marked with a status of TRUST and that it is stored in your server key database. During the SSL handshake the server tells the client which CA's it trusts based on the trusted CA's in the server key database. The browser then searches its client certificates for ones issued by these CA's and allows the user to choose from a list which certificate to send to the server. This example assumes that you are using the same CA to sign your client certificate that you used to sign your server certificate. This implies that the CA certificate is already in your server key database and marked with a status of TRUST.

Note: If you run multiple Web servers, make sure that you store the server, Certificate Authority (CA), and client key databases for a particular Web server in a separate set of files from key databases belonging to other Web servers. Otherwise, if one of the files becomes corrupted, you lose key

databases for multiple Web servers. SSL does not initialize for any of the Web servers whose server key databases are stored in a corrupted file.

RACF notes:

- The user ID issuing the RACF commands must have the proper authority.
- Choose the environment in which you execute the RACF commands (TSO READY, ISPF option 6, and so forth). Implementing these commands varies from one environment to another. See your local RACF administrator for assistance, or review the appropriate books for your environment. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Related information: The Web server provides support for retrieving security information from a Lightweight Directory Access Protocol (LDAP) server. For more information on LDAP support, see your Chapter 15, "Retrieving Lightweight Directory Access Protocol information" on page 297.

Summary of tasks

In this example, you will perform the following main tasks:

1. Update server configuration directives
2. Insure the CA certificate is in the client's browser
3. If acting as your own CA, provide option(s) for obtaining client certificates
4. Map a client certificate to RACF
5. Verify that the client can access a protected page with the client certificate

Step 1. Update server configuration directives

SSLClientAuth directive: The default setting for the SSLClientAuth directive is off. Because the server uses client certificates to validate clients and checks for CA certificates in the local database, specify:

```
SSLClientAuth  local
```

UserId directive:

Note: UserId directive refers to the default access control user ID for the Web server. UserId subdirective refers to the UserId specified on a Protection directive or protection setup. A UserId setting on a protection setup overrides the user ID specified on the UserId directive.

The initial configuration file setting for the UserId directive is:

```
UserId  %%CLIENT%%
```

In this example, %%CLIENT%% is used for the default access control user ID; %%CERTIF%% is generally used only if your entire Web site is protected. In this example, information in a specific directory is protected using client certificates, not the entire Web site. Therefore, %%CERTIF%% is specified only on the UserId subdirective on the Protection directive.

For more information on the UserId directive , see "Userid - Specify the Default Access Control user ID" on page 457.

Protection and Protect directives: For this example, add the following Protection and Protect directives to the server configuration file:

```
Protection PROTECTED_INFO {
       ServerId        Security_Administration
       AuthType        Basic
       PasswdFile      %%SAF%%
       UserId          %%CERTIF%%
       Mask            anybody
}

Protect  /secret/*  PROTECTED_INFO
```

Notes:

1. **ServerId**: Specify a name of your choice. Make this name unique for each protection setup.
2. **AuthType**: The AuthType subdirective lets you limit access based on user names and passwords. With an AuthType of Basic, passwords are sent to the client as plain text; they are encoded but not encrypted.
3. **PasswdFile**: For client certificates, the value of this field must be `%%SAF%%`.
4. **UserId**: For client certificates, the value of this field must be `%%CERTIF%%`. By specifying `%%CERTIF%%` on the UserId subdirective of the Protection directive, you indicate to use client certificates to control access to information only in the directory specified on the Protect directive. If you specify `%%CERTIF%%` on the UserId directive, client certificates are used to control access to all information on the Web site.
5. **Mask**: For client certificates, the value of the field must be one of the following:
   - `@`
   - `anybody`
   - `anyone`
   - `anonymous`
   - `anybody@(*)`
   - `anyone@(*)`
   - `anonymous@(*)`

   By using one of these values, the server completes requests without prompting for a user ID and password, as long as the client certificate maps to a RACF user ID. However, if the client certificate is not associated with a RACF user ID, the server asks for the user ID and password. The browser pop-up window for the user ID and password contains the words `System_Logon` instead of the value on the `ServerID` subdirective.
6. The Protect directive must specify the name of the resource or directory that you are protecting, along with a reference to the Protection directive that defines the type of protection provided. The Protect directive in this example references the PROTECTED_INFO Protection directive. The information in the `/secret/` directory is protected using client certificates. Note that password protection is still the default for accessing protected information; a client who does not have a client certificate is prompted for a RACF user ID and password.
7. For more information on Protection subdirectives, see “Protection subdirectives” on page 444.

Step 2. Insure the CA certificate is in the client’s browser

Browsers vary as to whether they require the CA certificate that signs the client certificate to be in the browser. This example assumes you will insure that your CA certificate is added to your browser if it is not already there.

If you are using an external commercial CA whose CA certificate is already loaded into the browser, you can skip this step and go to "Step 4. Map a client certificate to RACF" on page 119. If the external CA certificate is not in the browser, contact the CA to obtain the CA certificate. Follow the CA's directions for loading the certificate into the browser.

If you are acting as your own CA and you are using the same CA certificate to sign client certificates that you used to sign your server certificate, then clients may have already loaded the CA certificate into their browsers in "Step 5. Add your CA name to the list of trusted CAs for your browser" on page 112. If they have not, they can dowload your CA certificate by opening a browser and entering URL:

```
http://server1.raleigh.ibm.com/cert.arm
```

- `server1.raleigh.ibm.com` is the fully qualified host name of your server.
- `cert.arm` is the file containing your CA certificate.

Step 3. If acting as your own CA, provide option(s) for obtaining client certificates

If you are using an external commercial CA, such as VeriSign or Thawte, to obtain your client certificates, you can skip this step and go to "Step 4. Map a client certificate to RACF" on page 119.

If you act as your own CA, you can use the following options for providing client certificates. The steps to implement client authentication vary depending on the application.

- Create the client certificates yourself using the gskkyman utility, RACDCERT, or a CA software product such as the RACF PKISERV application, and send the signed certificates to clients who connect to your server.
- Request that clients generate their own certificate requests and send them to you for signature. This method requires that clients have access to the gskkyman utility or a CA software product such as the RACF PKISERV application.

To find out more about the RACF PKISERV application, go to URL: `http://www.ibm.com/s390/products/racf/goodies.html`.

You can use one or more of these options. This example shows steps for the first option using gskkyman.

Creating your client key database: To create your client key database:

1. Change to the path of the new client key database location by entering **cd** *path*. In this example, *path* is `/ibm/security/user1`.
2. Enter `gskkyman` to start the utility.
3. Enter **1** to create a new client key database.
4. Enter a unique name for your client key database name. In this example, `client.kdb` is used.
5. Enter a password using the guidelines in "Key database password" on page 158.
6. Enter the password again for verification.
7. Indicate whether the password should expire. If yes, accept the default of 60 days or enter the number of days. A message is displayed confirming that the database has been created.
8. Enter **1** to continue working with the client key database.

Creating a client certificate request: To create a client certificate:

1. Enter **3** to create a new key pair and certificate request.
2. Enter a client certificate request name. In this example, the certificate name is `client1.arm`. The certificate request file is in Base 64 encoded ASCII format. For more information on certificate file types, see "Certificate file types generated by the gskkyman utility" on page 157.
3. Enter a name or label to use for identifying the key and certificate in the database, for example, **Certificate for Jane Smith**. Avoid using the word `request` when specifying labels.
4. Enter the number of the key size you want to use. A smaller key size impacts performance less because it requires fewer resources to encrypt and decrypt than a larger key size. However, you need to determine the appropriate key size for your security needs. For more information on encryption and supported key sizes, see "Encryption support for the HTTP Server" on page 64.
5. Complete the fields that generate the Distinguished Name (DN):
   - Enter the common name of the client, for example, **Jane Smith**.
   - Enter an organization name, for example, **IBM ID Raleigh**.
   - Optionally enter an organization unit, for example, **IBM ID z/OS** .
   - Optionally enter a city or locality, for example, **Raleigh**.
   - Optionally enter a state or province, for example, **North Carolina**
   - Enter a country code, for example, **US**. Only 2 characters are allowed.

   A message displays confirming that the request has completed successfully. This message indicates that the gskkyman utility created the `client1.arm` file.
6. Enter **1** to exit the gskkyman utility.

Signing a client certificate: To sign the client certificate request using the file names in this example, enter the following command:

```
gskkyman -g -x 365 -cr client1.arm -ct client1.txt -k cakey.kdb
```

To issue this command, you must have access to the certificate request file `client1.arm` and the CA key database `cakey.kdb`. If these files are not in your current working directory, enter their fully qualified filenames on the command. You will also be prompted for the CA key database password.

- `-x 365` specifies that this certificate is valid for 365 days. This number must be less than the number of days the CA certificate is valid, which is `1500` in this example. If the client and CA certificates are not created on the same day, the effective number of days starts to decrement for the CA certificate.
- `client1.arm` is the certificate request file created by the client.
- `client1.txt` is the file created by this command. This file contains the signed client certificate. You specify a filename of your choice.
- `cakey.kdb` is the name of the CA key database you created in "Step 1. Set up the CA key database" on page 108.

Storing the CA certificate in the client key database: The CA certificate that signed your client certificate must first be stored in the client key database before the client certificate can be received into the database. To store the CA certificate:

1. Change to the path where the client key database is located by entering **cd** *path*. In this example, *path* is **/ibm/security/user1**.
2. Enter **gskkyman** to start the utility.
3. Enter **2** to open your client key database.
4. Enter the database name, `client.kdb`.
5. Enter the database password.
6. Enter **6** to store the CA certificate.

7. Enter the certificate file name. In this example, the file was created in "Creating your self-signed CA certificate" on page 108. The file name is `cert.arm`.
8. Enter the label for the certificate, **CA certificate for server1**.
9. Enter **0** to return to the key database menu.

Receiving the signed client certificate into your client key database: To receive the signed client certificate into your client key database:
1. Enter **4** to receive the client certificate.
2. Enter the certificate file name. In this example, the file name is `client1.txt`.
3. Do not designate the certificate as the default key in the client key database.
4. Enter **1** to exit gskkyman.

Sending the signed client certificate to the client:

Exporting the signed client certificate into a PKCS12 file: The signed client certificate must be exported into a PKCS12 file before loading it into your browser:
1. Enter **9** to export the certificate.
2. Enter **2** to export the certificate to a PKCS12 file.
3. Select the label of the signed client certificate in the list, **Certificate for Jane Smith**.
4. Enter the name of the PKCS12 format file that will contain the signed certificate, for example, `client1.p12`.
5. Enter a password using the guidelines in "Key database password" on page 158.
6. Enter the password again for verification.
7. Enter **1** to exit gskkyman.

Transferring the PKCS12 file to the client's workstation: This example shows how to use the FTP command to transfer the PKCS12 file containing the signed client certificate to the client's workstation. The following steps are performed on the workstation:
1. Enter the FTP command and the host name or IP address of the server, for example, **ftp server1.raleigh.ibm.com**.
2. When prompted, enter your user ID and password.
3. Change to the directory where the client certificate PKCS12 file `client1.p12` is located, for example, **cd /ibm/security/user1**.
4. Enter **bin** to transfer the file in binary format.
5. Transfer the file to the workstation by entering **get client1.p12**.
6. Type **quit** to exit.

Loading the PKCS12 file into the client's browser: This example shows how to load the PKCS12 file into the Netscape Communicator browser:
1. Start the browser.
2. **Click Communicator** → **Tools** → **Security Info**, to access the security information.
3. **Click Yours**, under **Certificates**.
4. **Click Import a Certificate**. You may need to scroll down to see this option.
5. Highlight the PKCS12 file.
6. **Click Open**, and enter the case sensitive password protecting the file.
7. **Click OK**. The following message displays: `Your certificates have been successfully imported.`

8. **Click OK**. You should see the certificate label in the window called **These are your certificates**. You may need to scroll down to find the label.

   Note: On browser versions prior to Netscape 4.6.1, there can be a problem displaying the label, for example, the label name may appear as **????@????**.

Step 4. Map a client certificate to RACF

You can use the following Resource Access Control Facility (RACF) options to map a client certificate to RACF when the client certificate is created with some method other than RACF commands or a RACF application such as PKISERV:

- **Certificate Name Filtering function:** This function is available for OS/390 Release 10 and later.
- **Automatic registration of digital certificates on the Web:** The Autoregistration Web Application enables a client to automatically register a certificate with the Web server.
- **Using ISPF panels or the RACDCERT command:** These two options take the same inputs, but you use panels with ISPF and a command line with RACDCERT.

For information on these options, see the *z/OS Security Server (RACF): Security Administrator's Guide*. To access this guide and other RACF books on the Web, go to URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

You can use one or more of these options. This example shows you how to use the RACDCERT command.

In this example you will:

- Store the client certificate in an MVS dataset.
- Associate the client certificate with a RACF user ID.

Export the client certificate into a file: Beginning in OS/390 Release 8, RACF maps client certificates that are in PKCS12 format. Since some browsers require client certificates in PKCS12 format, we will map a PKCS12 formatted file to a RACF user ID. If you are acting as your own CA, we will use the same PKCS12 file sent to the client in "Sending the signed client certificate to the client" on page 118 (`client1.p12`) to map the client certificate to RACF. If you are using an external commercial CA, you can export the client certificate from the browser. The PKCS12 file contains the private key and certificate.

Note: The client certificate can be output from gskkyman in binary or base 64 encoded ASCII for input to RACF. Some browsers may also be able to output the client certificate in these formats or other formats for input to RACF. If either the binary or base 64 encoded ASCII format is outputted, the resulting file would contain the certificate without the private key. The format of the client certificate in RACF can be different from the format of the certificate in the browser.

Placing the client certificate file in an MVS data set: You must store client certificates on MVS in variable block (VB) format. The client certificates in this example are in an HFS directory. Use the TSO/E **OGET** command to move the certificate file from the HFS directory into an MVS sequential data set. If you move the certificate into a new data set, the **OGET** command creates a VB sequential data set by default. You must use the **OGET** command to move the client certificate into the MVS sequential data set; exporting it from the browser to FTP it directly into the MVS data set does not work because the certificate file is not in the correct format.

Example:

```
oget '/ibm/security/user1/client1.p12' 'jsmith.client1.p12' binary
```

Note: You can execute this TSO/E command from TSO/E, ISPF option 6, and the shell. To find out more about this command, including where to execute it, please see the *z/OS UNIX System Services Command Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Installing the client certificate in RACF: To perform the following steps, ensure that you are using an MVS user ID that has the authority to map a client certificate to an MVS user ID.

1. Issue the **RACDCERT** command to add the client certificate to the RACF data base for each client. For example:

   ```
   RACDCERT ID(RACFID1) ADD('JSMITH.CLIENT1.P12')
   WITHLABEL('Certificate for Jane Smith') TRUST PASSWORD('X2RL')
   ```

   - **RACFID1** is the RACF user ID under which the client certificate is added.
   - `'SERVER1.CLIENT1.P12'` is the name of the dataset where the certificate file is located.
   - **TRUST** indicates that you can use the client certificate to authenticate the user ID **RACFID1**.
   - **Certificate for Jane Smith** is the label of the client certificate.
   - **X2RL** is the required password for PKCS12 certificates.
2. Issue the following **SETROPTS** command to refresh the DIGTCERT class for all the clients:

   ```
   SETROPTS RACLIST(DIGTCERT) REFRESH
   ```

3. Verify that the client certificate is associated with a user ID that has been defined to RACF, by issuing the following **RACDCERT** command and specifying the user ID in the ID field:

   ```
   RACDCERT ID(RACFID1) LIST
   ```

   If you are logged onto the user ID you are trying to verify, you can issue the RACDCERT command without operands to display the certificate for the user ID.

Step 5. Verify that the client can access a protected page with the client certificate

To test that client authentication is set up correctly for the client in this example:

1. Start your Web server.
2. Start the browser and specify URL:

   `https://server1.raleigh.ibm.com/secret/budget.html`
3. Select the label for the client certificate, when prompted by the browser. If the setup is correct, you view the page without prompts for a user ID and password.

Example 5 for RACDCERT: Setting up secure connections using a self-signed server certificate

This example is a quick start method for setting up SSL on your server.

Note: Using a self-signed server certificate is recommended only for test environments. This method gives you a quick way to test your SSL setup and internal applications with a small number of trusted clients.

System SSL notes:

You must use System SSL to establish secure connections. To use System SSL with the Web server, the System SSL load library must exist in linklist and must be under program control. If you have not already done so:

- Add the load library to the linklist.
- Turn on program control for the library by issuing the following RACF commands from a user ID that has the proper authority:

```
RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

  If turning on program control for the first time, use the RDEFINE command instead of the RALTER command.

RACF notes:

- The user ID issuing the RACF commands must have the proper authority.
- Choose the environment in which you execute the RACF commands (TSO READY, ISPF option 6, and so forth). Implementing these commands varies from one environment to another. See your local RACF administrator for assistance, or review the appropriate books for your environment. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Summary of tasks

In this example, you will perform the following main tasks:

1. Create the server key ring
2. Create your self-signed server certificate
3. Connect your signed server certificate to the server key ring
4. Permit the ID that is used to start the server to access to the key ring
5. Register the server key ring with the Web server
6. Use server security defaults
7. Verify that you can establish a secure connection with the server

Step 1. Create the server key ring

A server key ring is required for each server that clients connect to using a secure SSL connection. Specify the key ring name on the KeyFile directive in the server configuration file, `httpd.conf`.

Assign server certificates and server key rings to the same user ID used to start the Web server. This action is required for SSL to initialize.To ensure you use the correct user ID, check the Web server header in the job log or vv trace, and find the line beginning with **Running as**. The Web server user ID appears in quotes. In the following example WEBSRV represents the user ID used to start the Web server:

```
............ This is IBM HTTP Server V5R3M0
............ Built on Sep  5 2000 at 09:58:01.
............ Started at Thu Sep  7 15:18:48 2000
............ Running as "WEBSRV", UID:0, GID:205.
```

In this step, you will create:

- **SERVA**, the server key ring.

To create the server key ring for this example, issue the RACF command:

```
RACDCERT ID(WEBSRV) ADDRING(SERVA)
```

Step 2. Create your self-signed server certificate

In this step you will:

- Create a server certificate with label **Certificate for user1**.

To generate the self-signed server certificate for this example, issue:

```
RACDCERT ID(WEBSRV) GENCERT SUBJECTSDN(CN('serva.raleigh.ibm.com')
O('IBM ID Raleigh') ou('IBM ID z/OS') L('Raleigh') SP('North Carolina') C('US'))
SIZE(512) WITHLABEL('Certificate for serva')
```

where

- the Distinguished Name consists of the:
  - Common name (Domain Name), **serva.raleigh.ibm.com**.
  - Organization name, **IBM ID Raleigh**.
  - Optional organizational unit, **IBM ID z/OS**.
  - Optional city or locality, **Raleigh**.
  - Optional state or province, **North Carolina**.
  - Country code, **US**.
- **WEBSRV** is the ID used to start the Web server and the ID under which the server key ring and server certificate reside.
- **512** is the key size.
- **Certificate for serva** is the label used to identify the key and certificate in the key ring.

Step 3: Connect your signed server certificate to the key ring

In this step you will:

- Connect the server certificate to the key ring.
- Insure that the server certificate is the default in the key ring.

To connect the server certificate that is now in RACF to your **SERVA** key ring and insure that the certificate will be the default in the key ring, issue:

```
RACDCERT ID(WEBSRV) CONNECT(ID(WEBSRV) LABEL('Certificate for serva')
RING(SERVA) DEFAULT)
```

where

- **WEBSRV** is the ID used to start the Web server and the ID to which the server certificate is connected.
- **SERVA** is the server key ring.
- **Certificate for serva** is the label used to identify the key and certificate in the key ring.
- **DEFAULT** makes the server certificate the default in the key ring.

Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.

Step 4: Permit the Web server ID to access the key ring through DIGTCERT

In this step you permit the user ID **WEBSRV** to access the key ring through the **DIGTCERT** general resource class.

The ID used to start the Web server must have access to the key ring created using RACDCERT. If the ID does not have access, SSL initialization fails. In this example the Web server startup ID is WEBSRV. To permit WEBSRV to access the server key ring, you issue RACF commands to perform the following tasks:

- Define the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING resources with universal access of `None`.
- Permit the WEBSRV ID read access to the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING resources in the FACILITY class.
- Activate the FACILITY general resource class.
- Refresh the FACILITY general resource class.

To perform these tasks, issue the following commands:

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
PE IRR.DIGTCERT.LIST CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)

RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PE IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)

SETR CLASSACT(FACILITY)
SETR RACLIST(FACILITY) REFRESH
```

To find out more about controlling access to the RACDCERT function through the FACILITY general resource class, see the *z/OS Security Server (RACF): Security Administrator's Guide* and the description of the RACDCERT command in the *z/OS Security Server (RACF): Security Administrator's Guide*. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Step 5. Register the server key ring with the Web server

To register the server key ring in this example with the Web server, add the following KeyFile directive to your server configuration file, httpd.conf:

```
KeyFile SERVA SAF
```

Step 6. Use server security defaults

In this example, the following security default settings are used:

SSLMode
: Support for SSL connections is set **on**.

SSLPort
: Port 443 is the default port used for SSL.

SSLCipherSpec
: The server uses an encryption level, supported by both the client and the server.

Step 7. Verify that you can establish a secure connection with the server

To verify that you can establish a secure connection with your server, start your Web server. Then start a browser and enter the following URL:

```
https://your_server_name:SSL_port_number
```

The URL for this example is:

```
https://serva.raleigh.ibm.com:443
```

Example 6 for RACDCERT: Setting up secure connections using certificates signed by an external commercial CA

This example includes steps for using an external commercial CA to sign your server certificate.

Note: Using an external commercial CA, such as VeriSign or Thawte, is recommended for Internet connections and e-business. This method provides the highest level of security. You can optionally use client authentication to verify the identity of those accessing your server.

System SSL notes:

You must use System SSL to establish secure connections. To use System SSL with the Web server, the System SSL load library must exist in linklist and must be under program control. If you have not already done so:

- Add the load library to the linklist.
- Turn on program control for the library by issuing the following RACF commands from a user ID that has the proper authority:

  ```
  RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
  SETROPTS WHEN(PROGRAM) REFRESH
  ```

  If turning on program control for the first time, use the RDEFINE command instead of the RALTER command.

RACF notes:

- The user ID issuing the RACF commands must have the proper authority.
- Choose the environment in which you execute the RACF commands (TSO READY, ISPF option 6, and so forth). Implementing these commands varies from one environment to another. See your local RACF administrator for assistance, or review the appropriate books for your environment. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Summary of tasks

In this example, you will perform the following main tasks:

1. Create the server key ring
2. Create your server certificate request
3. Send your server certificate request to an external commercial CA
4. Insure that your CA certificate is in RACF and marked with a status of TRUST
5. Add your server certificate to the key ring
6. Connect your signed server certificate to the server key ring
7. Permit the ID that is used to start the server to access to the key ring
8. Register the server key ring with the server
9. Use server security defaults
10. Verify that you can establish a secure connection with the server
11. Optionally, set up client authentication using client certificates

Step 1. Create the server key ring

A server key ring is required for each server that clients connect to using a secure SSL connection. Specify the key ring name on the KeyFile directive in the server configuration file, `httpd.conf`.

Assign server certificates and server key rings to the same user ID used to start the Web server. This action is required for SSL to initialize.To ensure you use the correct user ID, check the Web server header in the job log or vv trace, and find the line beginning with **Running as**. The Web server user ID appears in quotes. In the following example WEBSRV represents the user ID used to start the Web server:

```
........... This is IBM HTTP Server V5R3M0
........... Built on Sep  5 2000 at 09:58:01.
........... Started at Thu Sep  7 15:18:48 2000
........... Running as "WEBSRV", UID:0, GID:205.
```

In this step, you will create:

- **SERVER1**, the server key ring.

To create the server key ring, **SERVER1**, issue the RACF command:

```
RACDCERT ID(WEBSRV) ADDRING(SERVER1)
```

Step 2. Create your server certificate request

In this step you will:

- Create a self-signed server certificate in order to establish your common name (host name) and public-private key pair.
- Generate a server certificate request from the self-signed certificate and save it to a data set.

The self-signed certificate that you create will contain the server's common name and public-private key pair. This information is required in order to generate the certificate request and obtain a certificate signed by your external CA. To create the self-signed certificate, issue:

```
RACDCERT ID(WEBSRV) GENCERT SUBJECTSDN(CN('server1.raleigh.ibm.com')
o('IBM ID Raleigh') ou('IBM ID z/OS') L('Raleigh') SP('North Carolina')
C('US')) SIZE(512) WITHLABEL('Certificate for server1')
```

where

- The Distinguished Name includes:
  - Common name (Domain Name), **server1.raleigh.ibm.com**.
  - Organization name, **IBM ID Raleigh**.
  - Optional organizational unit, **IBM ID z/OS**.
  - Optional city or locality, **Raleigh**.
  - Optional state or province, **North Carolina**.
  - Country code, **US**.
- **WEBSRV** is the ID used to start the Web server and the ID under which the server certificate is created.
- **512** is the key size.
- **Certificate for server1** is the label of the server certificate request.

To generate a server certificate request and save it to a data set , issue:

```
RACDCERT ID(WEBSRV) GENREQ(LABEL('Certificate for server1')) DSN(CERTREQ.ARM)
```

where

- **WEBSRV** is the ID used to start the Web server and the ID under which the server certificate resides.
- **Certificate for server1** is the label for the server certificate request.
- `CERTREQ.ARM` is the data set which contains your server public-private key pair and unsigned certificate.

Step 3. Send your server certificate request to an external commercial CA

After you create your server certificate request, you send that request to a certificate authority (CA) to sign. First transfer the data set containing the server certificate request to your workstation. Then follow the instructions from the CA for sending and receiving your certificate request.

Transferring the server certificate request to your workstation: This example shows how to use the File Transfer Protocol (FTP) command to transfer the `CERTREQ.ARM` data set containing the server certificate request to your workstation. Perform the following steps on the workstation:

1. Enter the **FTP** command and either the host name or the IP address of the server. For example, enter `ftp server1.raleigh.ibm.com`.
2. Enter your user ID and password when prompted.
3. Change to the high level qualifier where you put the data set containing the server certificate request, `CERTREQ.ARM`. For example, enter `cd 'USER1'`.
4. Transfer the data set to the workstation in ASCII format by entering `get CERTREQ.ARM`.
5. Type **quit** to exit.

Step 4. Insure that your CA certificate is in RACF and marked with a status of TRUST

Your CA certificate must be in the RACF list of CA certificates and must be marked with a status of **TRUST** before you can receive the CA-signed certificate into your server key ring.

By default, the following CAs are designated in RACF, with a status of **NO TRUST**. Before you can use one of these CAs in the list, you must first mark that CA with a status of **TRUST**.

- Integrion Certification Authority Root
- IBM World Registry Certification Authority
- Thawte Personal Premium CA
- Thawte Personal Freemail CA
- Thawte Personal Basic CA
- Thawte Premium Server CA
- Thawte Server CA
- RSA Secure Server Certification Authority
- VeriSign Class 1 Public Primary Certification Authority
- VeriSign Class 2 Public Primary Certification Authority
- VeriSign Class 3 Public Primary Certification Authority

In this step, you will:

- Check whether your external CA is in the list of CAs in RACF and whether it is marked with a status of **TRUST**.
- Add the CA certificate to RACF and mark it with a status of **TRUST** if it is not in the list of CAs in RACF.

- If the CA certificate is in the list of CAs in RACF, but the status is **NO TRUST**, then change the status to **TRUST**.
- Connect the CA certificate to your **SERVER1** key ring.

To check the list of CAs, issue:

```
RACDCERT CERTAUTH
```

After receiving the CA certificate, add the certificate to RACF with **TRUST** status. For example, issue the following RACF command:

```
RACDCERT CERTAUTH ADD(CERT1.ARM) TRUST WITHLABEL('CA cert for server1')
```

where

- **CERTAUTH** indicates the type of certificate you are adding to RACF, in this case, a certificate authority certificate.
- **CERT1.ARM** is the data set containing the CA certificate.
- **CA cert for server1** is the label for the CA certificate.

If your CA is in the list of CAs in RACF, but has a current status of **NO TRUST**, then change the status to **TRUST**. For example, issue the following command:

```
RACDCERT CERTAUTH ALTER(LABEL('CA cert for server1')) TRUST
```

where:

- **CERTAUTH** indicates the type of certificate you are altering in RACF, in this case, a certificate authority certificate.
- **CA cert for server1** is the label for the CA certificate.

Now that your CA certificate is in the RACF list of CA certificates and has a status of trust, connect the CA certificate to your **SERVER1** key ring. To connect the CA certificate to the key ring, issue the following RACF command:

```
RACDCERT ID(WEBSRV) CONNECT(CERTAUTH LABEL('CA cert for server1') RING(SE
RVER1)
```

where:

- **WEBSRV** is the ID used to start the Web server and the ID under which the server key ring resides.
- **CA cert for server1** is the label for the CA certificate.
- **CERTAUTH** indicates the type of certificate connected, in this case, a certificate authority certificate.

Step 5. Add your server certificate to the key ring

In this step, you will:

- Alter the server certificate if necessary to make it look like the example.
- Put the server certificate in an MVS data set, **SERVCERT.ARM**.
- Add the server certificate to RACF and associate it with the**WEBSRV** ID.

Before you can add the signed server certificate to your server key ring, you must put your server certificate in an MVS data set. The certificate data set you create in this example is `SERVCERT.ARM`. Alter the certificate data set if necessary. Include the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines and all data inbetween as shown in the following example. If your certificate contains additional information before `BEGIN CERTIFICATE` or after `END CERTIFICATE`, remove all of the extraneous information in the file.

```
-----BEGIN CERTIFICATE-----
 MIIB0DCCATkCBDV8PgswDQYJKoZIhvcNAQECBQAwcjELMAkGA1UEBhMCVVMxDTAL
 BgNVBAgTBE4uQy4xDDAKBgNVBAcTA1JUUDEMMAoGA1UEChMDSUJNMRcwFQYDVQQL
 Ew5XZWJzZXJ2ZXIgVGVzdDEfMB0GA1UEAxMWbXZzMTY3LnJhbGVpZ2guaWJtLmNv
 bTAaFws5ODA2MDgxOTM5WhcLOTkwNjA4MTkzOVowNDELMAkGA1UEBhMCVVMxDDAK
 BgNVBAoTA0lCTTEXMBUGA1UEAxMOcGtjczEwLmlibS5jb20wXDANBgkqhkiG9w0B
 AQEFAANLADBIAkEA1IYGldVmnKAI8hJQGT074oXTD0Tb+jFN8wkPqc+DVhYix1fj
 h/sbiuDZF66BMh5hnHfJr75633CgjW10EpID0wIDAQABMA0GCSqGSIb3DQEBAgUA
 A4GBAF1KVppAM7Gh2F9BBIY/jPMFlRp8+HAAVkK29Q4DxeF2FrTzQutKmO8duCWv
 xnJo4pgl5Uj29DSAsrX8mULfczyuZwVVXiCGnhN03pYj8bbQjo0edqQ7hYsRl3P4
 C72I+yRwtWUukfVgwdo0mWXyEclx7eT5jsW4weVEqWvuht8j
-----END CERTIFICATE-----
```


This example assumes that you received the server certificate onto your workstation. You can FTP the server certificate in ASCII format to an MVS data set. The following steps are performed on the workstation:

1. Enter the FTP command and the host name or IP address of the server, for example, **ftp server1.raleigh.ibm.com**.
2. When prompted, enter your user ID and password.
3. Transfer the file to an MVS data set that will be created on execution of the put command by entering **put servcert.arm 'USER1.SERVCERT.ARM'**.
4. Type **quit** to exit.

The server certificate signed by your external CA must be added to RACF and associated with the **WEBSRV** ID. In doing so, you will replace the self-signed certificate created in "Step 2. Create your server certificate request" on page 125. Issue:

```
RACDCERT ID(websrv) ADD(SERVCERT.ARM) WITHLABEL('Certificate for server1')
```

where

- **SERVCERT.ARM** is the server certificate data set.
- **Certificate for server1** is the label of the server certificate

Step 6: Connect your signed server certificate to the server key ring

In this step, you will:

- Connect the server certificate to your **SERVER1** key ring.
- Insure the server certificate will be the default in the server key ring.

Connect the server certificate that is now in RACF to your **SERVER1** key ring and make the certificate the default certificate in the key ring. For example, issue the following RACF command:

```
RACDCERT ID(WEBSRV) CONNECT(ID(WEBSRV) LABEL('Certificate for server1')
RING(SERVER1) DEFAULT)
```

where:

- **WEBSRV** is the ID used to start the Web server and the ID under which the server key ring and the server certificate reside.
- **SERVER1** is the server key ring.
- **Certificate for server1** is the label that identifies the key and certificate in the key ring.
- **DEFAULT** makes the server certificate the default in the key ring.

Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.

Step 7. Permit the Web server ID to access the key ring through DIGTCERT

In this step you permit the user ID **WEBSRV** to access the key ring through the **DIGTCERT** general resource class.

The ID used to start the Web server must have access to the key ring created using RACDCERT. If the ID does not have access, SSL initialization fails. In this example the Web server startup ID is WEBSRV. To permit WEBSRV to access the server key ring, you issue RACF commands to perform the following tasks:

- Define the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING resources with universal access of `None`.
- Permit the WEBSRV ID read access to the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING resources in the FACILITY class.
- Activate the FACILITY general resource class.
- Refresh the FACILITY general resource class.

To perform these tasks, issue the following commands:

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
PE IRR.DIGTCERT.LIST CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)

RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PE IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)

SETR CLASSACT(FACILITY)
SETR RACLIST(FACILITY) REFRESH
```

To find out more about controlling access to the RACDCERT function through the FACILITY general resource class, see the *z/OS Security Server (RACF): Security Administrator's Guide* and the description of the RACDCERT command in the *z/OS Security Server (RACF): Security Administrator's Guide*. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Step 8. Register the server key ring with the server

To register your server key ring in this example with the Web server, add the following KeyFile directive to your server configuration file, `httpd.conf`:

```
KeyFile SERVER1 SAF
```

Step 9. Use server security defaults

In this example, the following security default settings are used:

SSLMode
: Support for SSL connections is set **on**.

SSLPort
: Port 443 is the default port used for SSL.

SSLCipherSpec
: The server uses an encryption level, supported by both the client and the server.

You can change the Web server security defaults, by editing the appropriate security directives in the configuration file.

Step 10. Verify that you can establish a secure connection with the server

To verify that you can establish a secure connection with your server, start your Web server. Then start a browser and enter the following URL:

```
https://your_server_name:SSL_port_number
```

The URL for this example is:

```
https://serva.raleigh.ibm.com:443
```

Step 11. Optionally, set up client authentication using client certificates

For instructions, see "Example 8 for RACDCERT: Setting up client authentication using client certificates signed by an internal CA" on page 135 or "Example 9 for RACDCERT: Setting up client authentication using client certificates signed by an external CA" on page 140.

Example 7 for RACDCERT: Setting up secure connections using certificates signed by an internal CA

This example includes steps for setting up and administering your own certificate authority.

Note: Acting as your own CA is recommended only for test environments and private intranets. With this method, you set up your own CA and sign certificates. You can optionally use client authentication to verify the identity of those accessing your server.

System SSL notes:

You must use System SSL to establish secure connections. To use System SSL with the Web server, the System SSL load library must exist in linklist and must be under program control. If you have not already done so:

- Add the load library to the linklist.
- Turn on program control for the library by issuing the following RACF commands from a user ID that has the proper authority:

  ```
  RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
  SETROPTS WHEN(PROGRAM) REFRESH
  ```

  If turning on program control for the first time, use the RDEFINE command instead of the RALTER command.

RACF notes:

- The user ID issuing the RACF commands must have the proper authority.
- Choose the environment in which you execute the RACF commands (TSO READY, ISPF option 6, and so forth). Implementing these commands varies from one environment to another. See your local RACF administrator for assistance, or review the appropriate books for your environment. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Summary of tasks

In this example, you will perform the following main tasks:

1. Create a self-signed CA certificate
2. Set up the server key ring
3. Create and sign your server certificate
4. Connect your signed server certificate to the server key ring
5. Add your CA name to the list of trusted CAs for your browser
6. Permit the ID that is used to start the server to access to the key ring
7. Register the server key ring with the Web server
8. Use server security defaults
9. Verify that you can establish a secure connection with the server
10. Optionally, set up client authentication using client certificates

Step1. Create a self-signed CA certificate

In this step you will:

- Create a self-signed CA certificate with label **CA certificate for server1**.
- Create an ASCII MVS data set, `CERT.ARM`, which contains your CA public-private key pair and self-signed CA certificate.

To create your self-signed CA certificate, issue the RACF command:

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('IBM Raleigh z/OS ID server CA')
o('IBM ID Raleigh') ou('IBM ID z/OS') L('Raleigh') SP('North Carolina')
C('US')) SIZE(512) WITHLABEL('CA certificate for server1') NOTBEFORE(DATE(2000
-09-04)) NOTAFTER(DATE(2004-10-12))
```

where

- the Distinguished Name consists of the:
  - Common name (Domain Name), **IBM Raleigh z/OS ID server CA**.
  - Organization name, **IBM ID Raleigh**.
  - Optional organizational unit, **IBM ID z/OS**.
  - Optional city or locality, **Raleigh**.
  - Optional state or province, **North Carolina**.
  - Country code, **US**.
- **CERTAUTH** indicates that a CA certificate is being generated.
- **512** is the key size.
- **CA certificate for server1** is the label of the CA certificate.
- **NOTBEFORE(DATE(2000 -09-04)) NOTAFTER(DATE(2004-10-12))** indicates that the ceritificate is valid for 1500 days from September 4, 2000 through October 12, 2004.

To export the CA certificate to a data set so that the CA can be added to the list of trusted CAs on the browser in "Step 5. Add your CA name to the list of trusted CAs for your browser" on page 133, issue:

```
RACDCERT CERTAUTH EXPORT(LABEL('CA certificate for server1')) DSN(CERT.ARM)
FORMAT(CERTB64)
```

where

- **CERTAUTH** indicates that a CA certificate is being exported.
- **CA certificate for server1** is the label of the CA certificate.
- `CERT.ARM` is the data set that will contain the CA certificate.
- `CERTB64` indicates that the CA certificate is saved to the data set in BASE 64 encoded ASCII.

Step 2. Set up the server key ring

A server key ring is required for each server that clients connect to using a secure SSL connection. Specify the key ring name on the KeyFile directive in the server configuration file, `httpd.conf`.

Assign server certificates and server key rings to the same user ID used to start the Web server. This action is required for SSL to initialize.To ensure you use the correct user ID, check the Web server header in the job log or vv trace, and find the line beginning with **Running as**. The Web server user ID appears in quotes. In the following example WEBSRV represents the user ID used to start the Web server:

```
........... This is IBM HTTP Server V5R3M0
........... Built on Sep  5 2000 at 09:58:01.
........... Started at Thu Sep  7 15:18:48 2000
........... Running as "WEBSRV", UID:0, GID:205.
```

In this step you will:

- Create a server key ring.
- Connect your CA certificate to the server key ring.

To create the server key ring, **SERVER1**, issue the RACF command:

```
RACDCERT ID(WEBSRV) ADDRING(SERVER1)
```

To connect the CA certificate to the server key ring, issue:

```
RACDCERT ID(WEBSRV) CONNECT(CERTAUTH LABEL('CA certificate for server1') RING(SE
RVER1)
```

where

- **CERTAUTH** indicates that a CA certificate is being connected.
- **CA certificate for server1** is the label of the CA certificate.
- **SERVER1** is the server key ring.
- **WEBSRV** is the ID used to start the Web server and the ID under which the SERVER1 key ring resides.

Step 3. Create and sign your server certificate

In “Step1. Create a self-signed CA certificate” on page 131, you set up your CA environment which enables you to act as your own CA and sign certificates. A signed server certificate is required before clients can establish an SSL connection to your server. Because you are acting as your own CA, you will sign the server certificate that you create in this step. If you were using an external commercial CA, such as VeriSign, you would send the server certificate request to that CA for signature.

In this step you will:

- Create a server certificate with label **Certificate for server1** signed by the internal CA using label **CA certificate for server1**.

To create the server certificate signed by the internal CA, issue:

```
RACDCERT ID(WEBSRV) GENCERT SUBJECTSDN(CN('server1.raleigh.ibm.com')
O('IBM ID Raleigh') ou('IBM ID z/OS') L('Raleigh') SP('North Carolina')
C('US')) SIZE(512) WITHLABEL('Certificate for server1')
SIGNWITH(CERTAUTH LABEL('CA certificate for server1'))
```

where

- The Distinguished Name includes:

  - Common name (Domain Name), **server1.raleigh.ibm.com**.
  - Organization name, **IBM ID Raleigh**.
  - Optional organizational unit, **IBM ID z/OS**.
  - Optional city or locality, **Raleigh**.
  - Optional state or province, **North Carolina**.
  - Country code, **US**.
- **WEBSRV** is the ID used to start the Web server and the ID under which the server certificate is created.
- **512** is the key size.
- **Certificate for server1** is the label of the server certificate request.
- **CA certificate for server1** is the label of the CA certificate that is used to sign the server certificate.

Step 4. Connect your signed server certificate to the server key ring

In this step you will:

- Connect a server certificate with label **Certificate for server1** to the server key ring.
- Insure the server certificate will be the default in the server key ring.

Connect the server certificate that is now in RACF to your **SERVER1** key ring and make the certificate the default certificate in the key ring. For example, issue the following RACF command:

```
RACDCERT ID(WEBSRV) CONNECT(ID(WEBSRV) LABEL('Certificate for server1')
RING(SERVER1) DEFAULT)
```

where:

- **WEBSRV** is the ID used to start the Web server and the ID under which the server key ring and the server certificate reside.
- **SERVER1** is the server key ring.
- **Certificate for server1** is the label that identifies the key and certificate in the key ring.
- **DEFAULT** makes the server certificate the default in the key ring.

Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.

Step 5. Add your CA name to the list of trusted CAs for your browser

You need to add your CA name to the list of CAs that are known by your browser. To do this, download the **cert.arm** file you created in "Step1. Create a self-signed CA certificate" on page 131 to your browser.

Before downloading the **cert.arm** file:

- Copy the **CERT.ARM** data set from MVS to your document root directory in the HFS using the TSO/E OPUT command in MVS. The document root directory is on the Pass /* directive in your server configuration file. The default directory is used in this example:

  ```
  oput 'USER1.CERT.ARM' '/usr/lpp/internet/server_root/pub/cert.arm'
  ```

Note: If you put the `cert.arm` file in another directory, you need to add a Pass directive for the directory to the server configuration file if one does not already exist. Alter the URL to download the CA certificate to reflect the directory.

Note: You can execute this TSO/E command from TSO/E, ISPF option 6, and the shell. To find out more about this command, including where to execute it, please see the *z/OS UNIX System Services Command Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Download the CA certificate that uses the names in this example:

- Start your Web server.
- Open a browser.
- Enter the following URL:

  `http://server1.raleigh.ibm.com/cert.arm`

  – `server1.raleigh.ibm.com` is the fully qualified host name of the server.
  – `cert.arm` is the CA certificate file.

Follow the browser prompts to install the CA certificate. Newer versions of the Netscape and Microsoft Internet Explorer browsers automatically start a wizard to help you install the certificate. If you use Microsoft Internet Explorer, you may need to open the file rather than saving it to disk to start the wizard. See the online help in your browser or browser documentation for additional information. Generally for Netscape, if you receive a window asking if you would like to accept the CA to certify network sites, electronic mail (e-mail) users, and software developers, do so.

Step 6. Permit the Web server ID to access the key ring through DIGTCERT

In this step you permit the user ID **WEBSRV** to access the key ring through the **DIGTCERT** general resource class.

The ID used to start the Web server must have access to the key ring created using RACDCERT. If the ID does not have access, SSL initialization fails. In this example the Web server startup ID is WEBSRV. To permit WEBSRV to access the server key ring, you issue RACF commands to perform the following tasks:

- Define the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING resources with universal access of `None`.
- Permit the WEBSRV ID read access to the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING resources in the FACILITY class.
- Activate the FACILITY general resource class.
- Refresh the FACILITY general resource class.

To perform these tasks, issue the following commands:

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
PE IRR.DIGTCERT.LIST CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)

RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PE IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)

SETR CLASSACT(FACILITY)
SETR RACLIST(FACILITY) REFRESH
```

To find out more about controlling access to the RACDCERT function through the FACILITY general resource class, see the *z/OS Security Server (RACF): Security Administrator's Guide* and the description of the RACDCERT command in the *z/OS Security Server (RACF): Security Administrator's Guide*. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Step 7. Register the server key ring with the Web server

To register your server key ring in this example with the Web server, add the following KeyFile directive to your server configuration file, `httpd.conf`:

```
KeyFile SERVER1 SAF
```

Step 8. Use server security defaults

In this example, the following security default settings are used:

SSLMode
: Support for SSL connections is set **on**.

SSLPort
: Port 443 is the default port used for SSL.

SSLCipherSpec
: The server uses an encryption level, supported by both the client and the server.

You can change the Web server security defaults, by editing the appropriate security directives in the configuration file.

Step 9. Verify that you can establish a secure connection with the server

To verify that you can establish a secure connection with your server, start your Web server. Then start a browser and enter the following URL:

```
https://your_server_name:SSL_port_number
```

The URL for this example is:

```
https://serva.raleigh.ibm.com:443
```

Step 10. Optionally, set up client authentication

For instructions, see "Example 8 for RACDCERT: Setting up client authentication using client certificates signed by an internal CA" or "Example 9 for RACDCERT: Setting up client authentication using client certificates signed by an external CA" on page 140.

Example 8 for RACDCERT: Setting up client authentication using client certificates signed by an internal CA

This example shows how to set up client authentication using client certificates signed by an internal CA. Using an internal CA to sign your client certificates is independent of whether you used an internal or external CA to sign your server certificate.

- You must first set up secure connections using one of the following examples:
  - "Example 6 for RACDCERT: Setting up secure connections using certificates signed by an external commercial CA" on page 124
  - "Example 7 for RACDCERT: Setting up secure connections using certificates signed by an internal CA" on page 130
  - "Example 10: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1 Release 3 or earlier)" on page 151

- You must insure that the internal CA that signs your client certificates is marked with a status of TRUST and that it is connected to your server key ring. During the SSL handshake the server tells the client which CA's it trusts based on the trusted CA's in the server key ring. The browser then searches its client certificates for ones issued by these CA's and allows the user to choose which client certificate to send to the server.

  If you created and signed your server certificate using "Example 7 for RACDCERT: Setting up secure connections using certificates signed by an internal CA" on page 130, then you can use the internal CA defined in that example for the internal CA in this example. If you created and signed your server certificate using "Example 6 for RACDCERT: Setting up secure connections using certificates signed by an external commercial CA" on page 124, then you must set up an internal CA as described in "Step1. Create a self-signed CA certificate" on page 131 and connect the CA certificate to the server key ring as described in "Step 2. Set up the server key ring" on page 132 before proceeding.

  If you migrated your server certificate using "Example 10: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1 Release 3 or earlier)" on page 151, the CA can be an external CA or an internal CA. If it is an internal CA, then you can use that CA for the internal CA in this example. If it is an external CA, then you must set up an internal CA as described in "Step1. Create a self-signed CA certificate" on page 131 and connect the CA certificate to the server key ring as described in "Step 2. Set up the server key ring" on page 132 before proceeding.

Alternatives for client certificate protection: A variety of ways exists to set up protection for client certificates. This example shows how to map a client certificate to a RACF user ID. For alternatives, see "Creating protection setups for Secure Sockets Layer (SSL) client authentication" on page 168.

RACF notes:

- The user ID issuing the RACF commands must have the proper authority.
- Choose the environment in which you execute the RACF commands (TSO READY, ISPF option 6, and so forth). Implementing these commands varies from one environment to another. See your local RACF administrator for assistance, or review the appropriate books for your environment. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

If you act as your own CA, you can use the following options for providing client certificates. The steps to implement client authentication vary depending on the application.

- Create the client certificates yourself using the gskkyman utility, RACDCERT, or a CA software product such as the RACF PKISERV application, and send the signed certificates to clients who connect to your server.
- Request that clients generate their own certificate requests and send them to you for signature. This method requires that clients have access to the gskkyman utility or a CA software product such as the RACF PKISERV application.

To find out more about the RACF PKISERV application, go to URL: `http://www.ibm.com/s390/products/racf/goodies.html`.

You can use one or more of these options. This example shows steps for the first option using RACDCERT.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Summary of tasks

In this example, you will perform the following main tasks:

1. Update server configuration directives
2. Insure the CA certificate is in the client's browser
3. Create the client certificate and associate it with a RACF user ID
4. Add the signed client certificate to the client's browser
5. Verify that the client can access a protected page with the client certificate

Step 1. Update server configuration directives

SSLClientAuth directive: The default setting for the SSLClientAuth directive is off. Because the server uses client certificates to validate clients and checks for CA certificates in the local database, specify:

```
SSLClientAuth  local
```

UserId directive:

Note: UserId directive refers to the default access control user ID for the Web server. UserId subdirective refers to the UserId specified on a Protection directive or protection setup. A UserId setting on a protection setup overrides the user ID specified on the UserId directive.

The initial configuration file setting for the UserId directive is:

```
UserId  %%CLIENT%%
```

In this example, %%CLIENT%% is used for the default access control user ID; %%CERTIF%% is generally used only if your entire Web site is protected. In this example, information in a specific directory is protected using client certificates, not the entire Web site. Therefore, %%CERTIF%% is specified only on the UserId subdirective on the Protection directive.

For more information on the UserId directive , see "Userid - Specify the Default Access Control user ID" on page 457.

Protection and Protect directives: For this example, add the following Protection and Protect directives to the server configuration file:

```
Protection PROTECTED_INFO {
       ServerId        Security_Administration
       AuthType        Basic
       PasswdFile      %%SAF%%
       UserId          %%CERTIF%%
       Mask            anybody
}

Protect  /secret/*  PROTECTED_INFO
```

Notes:

1. **ServerId**: Specify a name of your choice. Make this name unique for each protection setup.
2. **AuthType**: The AuthType subdirective lets you limit access based on user names and passwords. With an AuthType of Basic, passwords are sent to the client as plain text; they are encoded but not encrypted.

3. **PasswdFile**: For client certificates, the value of this field must be `%%SAF%%`.
4. **UserId**: For client certificates, the value of this field must be `%%CERTIF%%`. By specifying `%%CERTIF%%` on the UserId subdirective of the Protection directive, you indicate to use client certificates to control access to information only in the directory specified on the Protect directive. If you specify `%%CERTIF%%` on the UserId directive, client certificates are used to control access to all information on the Web site.
5. **Mask**: For client certificates, the value of the field must be one of the following:
   - `@`
   - `anybody`
   - `anyone`
   - `anonymous`
   - `anybody@(*)`
   - `anyone@(*)`
   - `anonymous@(*)`

   By using one of these values, the server completes requests without prompting for a user ID and password, as long as the client certificate maps to a RACF user ID. However, if the client certificate is not associated with a RACF user ID, the server asks for the user ID and password. The browser pop-up window for the user ID and password contains the words `System_Logon` instead of the value on the `ServerID` subdirective.
6. The Protect directive must specify the name of the resource or directory that you are protecting, along with a reference to the Protection directive that defines the type of protection provided. The Protect directive in this example references the PROTECTED_INFO Protection directive. The information in the `/secret/` directory is protected using client certificates. Note that password protection is still the default for accessing protected information; a client who does not have a client certificate is prompted for a RACF user ID and password.
7. For more information on Protection subdirectives, see "Protection subdirectives" on page 444.

Step 2. Insure the CA certificate is in the client's browser

Browsers vary as to whether they require the CA certificate that signs the client certificate to be in the browser. This example assumes you will insure that your CA certificate is added to your browser if it is not already there.

If you are using the same CA certificate to sign client certificates that you used to sign your server certificate, then clients may have already loaded the CA certificate into their browsers in "Step 5. Add your CA name to the list of trusted CAs for your browser" on page 133. If they have not, they can dowload your CA certificate by opening a browser and entering URL:

```
http://server1.raleigh.ibm.com/cert.arm
```

- `server1.raleigh.ibm.com` is the fully qualified host name of your server.
- `cert.arm` is the file containing your CA certificate.

Step 3. Create the client certificate and associate it with a RACF user ID

This example assumes that your CA certificate was added to the list of trusted CA's in your browser. You will generate a client certificate under a RACF user ID so that the client certificate can be used to authenticate the user ID.

In this step you will:

- Create the client certificate with label **Certificate for Jane Smith** signed by the internal CA using label **CA certificate for server1**. The certificate will be created under user ID **JSMITH**.

To create the client certificate signed by the internal CA, issue the RACF command:

```
RACDCERT ID(JSMITH) GENCERT SUBJECTSDN(CN('Jane Smith')
O('IBM ID Raleigh') ou('IBM ID z/OS') L('Raleigh') SP('North Carolina')
C('US')) SIZE(512) WITHLABEL('Certificate for Jane Smith')
SIGNWITH(CERTAUTH LABEL('CA certificate for server1'))
```

where

- the Distinguished Name consists of the:
  - Common name (Domain Name), **Jane Smith**.
  - Organization name, **IBM ID Raleigh**.
  - Optional organizational unit, **IBM ID z/OS**.
  - Optional city or locality, **Raleigh**.
  - Optional state or province, **North Carolina**.
  - Country code, **US**.
- **JSMITH** is the MVS user ID under which the client certificate is to be added.
- **512** is the key size.
- **Certificate for Jane Smith** is the label of the client certificate.
- **CA certificate for server1** is the label of the CA certificate that will sign the client certificate.

The client certificate will be created with status **TRUST**. Trust indicates that the client certificate can be used to authenticate the user ID **JSMITH**.

Step 4. Add the signed client certificate to the client's browser

In this step you will:

- Export the client certificate to an MVS data set.
- FTP the client certificate to the client's workstation.
- Load the client certificate into the client's browser

Export the client certificate to an MVS data set: To export the client certificate to an MVS data set, issue:

```
RACDCERT ID(JSMITH) EXPORT(LABEL('Certificate for Jane Smith'))
DSN('JSMITH.CLIENT1.P12') FORMAT(PKCS12DER) PASSWORD('Test')
```

where

- **JSMITH** is the user ID associated with the client certificate to be exported.
- **Certificate for Jane Smith** is the label of the client certificate.
- **'JSMITH.CLIENT1.P12'** is the data set that will contain the client certificate.
- **PKCS12DER** indicates that the client certificate and private key are DER encoded when saved to the data set.
- **Test** is the password associated with the encrypted certificate.You will be required to provide this password when you import the client certificate into the browser. The password is case sensitive.

FTP the client certificate to the client's workstation: This example shows how to use the FTP command to transfer the PKCS12 data set containing the signed client certificate to the client's workstation. The following steps are performed on the workstation:

1. Enter the FTP command and the host name or IP address of the server, for example, **ftp server1.raleigh.ibm.com**.
2. When prompted, enter your user ID and password.
3. Enter **bin** to transfer the file in binary format.
4. Transfer the file to the workstation by entering **get 'JSMITH.CLIENT1.P12' client1.p12**.
5. Type **quit** to exit.

Load the client certificate into the client's browser: This example shows how to load the PKCS12 file into the Netscape Communicator browser:

1. Start the browser.
2. **Click Communicator** → **Tools** → **Security Info**, to access the security information.
3. **Click Yours**, under **Certificates**.
4. **Click Import a Certificate**. You may need to scroll down to see this option.
5. Highlight the PKCS12 file.
6. **Click Open**, and enter the case sensitive password protecting the file.
7. **Click OK**. The following message displays: `Your certificates have been successfully imported.`
8. **Click OK**. You should see the certificate label in the window called **These are your certificates**. You may need to scroll down to find the label.

   Note: On browser versions prior to Netscape 4.6.1, there can be a problem displaying the label, for example, the label name may appear as **????@????**.

Step 5. Verify that the client can access a protected page with the client certificate

To test that client authentication is set up correctly for the client in this example:

1. Start your Web server.
2. Start the browser and specify URL:
   `https://server1.raleigh.ibm.com/secret/budget.html`
3. Select the label for the client certificate, when prompted by the browser. If the setup is correct, you view the page without prompts for a user ID and password.

Example 9 for RACDCERT: Setting up client authentication using client certificates signed by an external CA

This example shows how to set up client authentication when client certificates are signed by an external CA.Using an external CA to sign your client certificates is independent of whether you used an internal or external CA to sign your server certificate. However, before using this example, there are a few things you must do first.

- You must set up secure connections using one of the following examples:
  - "Example 6 for RACDCERT: Setting up secure connections using certificates signed by an external commercial CA" on page 124
  - "Example 7 for RACDCERT: Setting up secure connections using certificates signed by an internal CA" on page 130
  - "Example 10: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1 Release 3 or earlier)" on page 151

- You must insure that the external CA that signs your client certificates is marked with a status of TRUST and that it is connected to your server key ring. During the SSL handshake the server tells the client which CAs it trusts based on the trusted CAs in the server key ring. The browser then searches its client certificates for ones issued by these CAs and allows the user to choose which client certificate to send to the server.

  If you created and signed your server certificate using "Example 6 for RACDCERT: Setting up secure connections using certificates signed by an external commercial CA" on page 124, then you can use the external CA defined in that example for the external CA in this example.

  If you created and signed your server certificate using "Example 7 for RACDCERT: Setting up secure connections using certificates signed by an internal CA" on page 130, then you must follow instructions in "Step 4. Insure that your CA certificate is in RACF and marked with a status of TRUST" on page 126 to insure that the external CA is in RACF with a status of TRUST, and that the external CA is connected to the server key ring.

  If you migrated your server certificate by using "Example 10: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1 Release 3 or earlier)" on page 151, you can use an external CA or an internal CA. If you use an external CA, then you can use that CA for the external CA in this example. If you use an internal CA, then follow instructions in "Step 4. Insure that your CA certificate is in RACF and marked with a status of TRUST" on page 126.

Alternatives for client certificate protection: A variety of ways exists to set up protection for client certificates. This example shows how to map a client certificate to a RACF user ID. For alternatives, see "Creating protection setups for Secure Sockets Layer (SSL) client authentication" on page 168.

RACF notes:

- The user ID issuing the RACF commands must have the proper authority.
- Choose the environment in which you execute the RACF commands (TSO READY, ISPF option 6, and so forth). Implementing these commands varies from one environment to another. See your local RACF administrator for assistance, or review the appropriate books for your environment. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Summary of tasks

In this example, you will perform the following main tasks:

1. Update server configuration directives
2. Insure the CA certificate is in the client's browser
3. Obtain client certificate
4. Map a client certificate to a RACF user ID
5. Verify that the client can access a protected page with the client certificate

Step 1. Update server configuration directives

SSLClientAuth directive: The default setting for the SSLClientAuth directive is off. Because the server uses client certificates to validate clients and checks for CA certificates in the local database, specify:

```
SSLClientAuth  local
```

UserId directive:

Note: UserId directive refers to the default access control user ID for the Web server. UserId subdirective refers to the UserId specified on a Protection directive or protection setup. A UserId setting on a protection setup overrides the user ID specified on the UserId directive.

The initial configuration file setting for the UserId directive is:

```
UserId  %%CLIENT%%
```

In this example, %%CLIENT%% is used for the default access control user ID; %%CERTIF%% is generally used only if your entire Web site is protected. In this example, information in a specific directory is protected using client certificates, not the entire Web site. Therefore, %%CERTIF%% is specified only on the UserId subdirective on the Protection directive.

For more information on the UserId directive , see "Userid - Specify the Default Access Control user ID" on page 457.

Protection and Protect directives: For this example, add the following Protection and Protect directives to the server configuration file:

```
Protection PROTECTED_INFO {
       ServerId        Security_Administration
       AuthType        Basic
       PasswdFile      %%SAF%%
       UserId          %%CERTIF%%
       Mask            anybody
}

Protect  /secret/*  PROTECTED_INFO
```

Notes:

1. **ServerId**: Specify a name of your choice. Make this name unique for each protection setup.
2. **AuthType**: The AuthType subdirective lets you limit access based on user names and passwords. With an AuthType of Basic, passwords are sent to the client as plain text; they are encoded but not encrypted.
3. **PasswdFile**: For client certificates, the value of this field must be `%%SAF%%`.
4. **UserId**: For client certificates, the value of this field must be `%%CERTIF%%`. By specifying `%%CERTIF%%` on the UserId subdirective of the Protection directive, you indicate to use client certificates to control access to information only in the directory specified on the Protect directive. If you specify `%%CERTIF%%` on the UserId directive, client certificates are used to control access to all information on the Web site.
5. **Mask**: For client certificates, the value of the field must be one of the following:
   - `@`
   - `anybody`
   - `anyone`

- `anonymous`
- `anybody@(*)`
- `anyone@(*)`
- `anonymous@(*)`

By using one of these values, the server completes requests without prompting for a user ID and password, as long as the client certificate maps to a RACF user ID. However, if the client certificate is not associated with a RACF user ID, the server asks for the user ID and password. The browser pop-up window for the user ID and password contains the words `System_Logon` instead of the value on the `ServerID` subdirective.

6. The Protect directive must specify the name of the resource or directory that you are protecting, along with a reference to the Protection directive that defines the type of protection provided. The Protect directive in this example references the PROTECTED_INFO Protection directive. The information in the `/secret/` directory is protected using client certificates. Note that password protection is still the default for accessing protected information; a client who does not have a client certificate is prompted for a RACF user ID and password.
7. For more information on Protection subdirectives, see "Protection subdirectives" on page 444.

Step 2. Insure the CA certificate is in the client's browser

Browsers vary as to whether they require the CA certificate that signs the client certificate to be in the browser. This example assumes you will insure that your CA certificate is added to your browser if it is not already there.

This example assumes you are using an external commercial CA. A number of commercial CA certificates may already be loaded in your clients' browsers, including the CA certificate that signed the client certificates. If it is loaded, proceed to the next step. If the external CA certificate is not in the browsers, contact the CA to obtain the CA certificate. Follow the CA's directions for loading the certificate into browsers.

Step 3. Obtain client certificate

Follow the external CA's instructions for obtaining the signed client certificate and loading it into your browser.

Step 4. Map a client certificate to a RACF user ID

RACF maps client certificates that are in various formats, including PKCS12, binary, and base 64 encoded ASCII. Some browsers may be able to output the client certificate in these formats or other formats. If, for instance, either the binary or base 64 encoded ASCII format is outputted, the resulting file would contain the client certificate without the private key. If the PKCS12 format is outputted, the resulting file would contain the private key (which RACF doesn't use) and the client certificate. If a browser does not output the client certificate in the format that you want, contact the signing authority to obtain the client certificate in the desired format. The format of the client certificate in RACF can be different from the format of the client certificate in the browser.

Since some browsers require client certificates in PKCS12 format, we will map a PKCS12 formatted certificate to a RACF user ID. You can export the client certificate from the browser so that it can be input to RACF.

Security

You can use the following Resource Access Control Facility (RACF) options to map a client certificate to RACF when the client certificate is created with some method other than RACF commands or a RACF application such as PKISERV:

- **Certificate Name Filtering function:** This function is available for OS/390 Release 10 and later.
- **Automatic registration of digital certificates on the Web:** The Autoregistration Web Application enables a client to automatically register a certificate with the Web server.
- **Using ISPF panels or the RACDCERT command:** These two options take the same inputs, but you use panels with ISPF and a command line with RACDCERT.

For information on these options, see the *z/OS Security Server (RACF): Security Administrator's Guide*. To access this guide and other RACF books on the Web, go to URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

You can use one or more of these options. This example shows how to use the RACDCERT command.

In this step you will:

- FTP the PKCS12 formatted client certificate from the workstation to the HFS on your z/OS system.
- Copy the PKCS12 formatted client certificate from the HFS to an MVS data set.
- Issue RACF commands to associate the client certificate with a RACF user ID.

FTP the client certificate from the client's workstation to the HFS: We will ftp the client certificate from the workstation into the HFS in this section and then do an OGET of the file into an MVS data set in the next section to insure that the data set containing the client certificate has the correct format. This example shows how to use the FTP command to transfer the signed PKCS12 formatted client certificate on a client's workstation to the HFS. The following steps are performed on the workstation:

1. Enter the FTP command and the host name or IP address of the server, for example, **ftp server1.raleigh.ibm.com**.
2. When prompted, enter your user ID and password.
3. Change to the directory where you will place the client certificate, for example, **cd /ibm/security/user1**.
4. Enter **bin** to transfer the file in binary format.
5. Transfer the file to the HFS by entering **put client1.p12**.
6. Type **quit** to exit.

Copy the client certificate from the HFS to an MVS data set: You must store client certificates on MVS in variable block (VB) format. The client certificates in this example are in an HFS directory. Use the TSO/E **OGET** command to move the certificate file from the HFS directory into an MVS sequential data set. If you move the certificate into a new data set, the **OGET** command creates a VB sequential data set by default. You must use the **OGET** command to move the client certificate into the MVS sequential data set; exporting it from the browser to FTP it directly into the MVS data set does not work because the certificate file is not in the correct format.

Example::

```
oget '/ibm/security/user1/client1.p12' 'jsmith.client1.p12' binary
```

Note: You can execute this TSO/E command from TSO/E, ISPF option 6, and the shell. To find out more about this command, including where to execute it, please see the *z/OS UNIX System Services Command Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Associate the client certificate with a RACF user ID: To perform the following steps, ensure that you are using an MVS user ID that has the authority to map a client certificate to an MVS user ID.

1. Issue the **RACDCERT** command to add the client certificate to the RACF data base for each client. For example:

   ```
   RACDCERT ID(RACFID1) ADD('JSMITH.CLIENT1.P12')
   WITHLABEL('Certificate for Jane Smith') TRUST PASSWORD('X2RL')
   ```

   - **RACFID1** is the RACF user ID under which the client certificate is added.
   - `'SERVER1.CLIENT1.P12'` is the name of the dataset where the certificate file is located.
   - **TRUST** indicates that you can use the client certificate to authenticate the user ID **RACFID1**.
   - **Certificate for Jane Smith** is the label of the client certificate.
   - **X2RL** is the required password for PKCS12 certificates.
2. Issue the following **SETROPTS** command to refresh the DIGTCERT class for all the clients:

   ```
   SETROPTS RACLIST(DIGTCERT) REFRESH
   ```

3. Verify that the client certificate is associated with a user ID that has been defined to RACF, by issuing the following **RACDCERT** command and specifying the user ID in the ID field:

   ```
   RACDCERT ID(RACFID1) LIST
   ```

   If you are logged onto the user ID you are trying to verify, you can issue the RACDCERT command without operands to display the certificate for the user ID.

Step 5. Verify that the client can access a protected page with the client certificate

To test that client authentication is set up correctly for the client in this example:

1. Start your Web server.
2. Start the browser and specify URL:

   `https://server1.raleigh.ibm.com/secret/budget.html`
3. Select the label for the client certificate, when prompted by the browser. If the setup is correct, you view the page without prompts for a user ID and password.

Example 10A: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1 Release 4 or later)

Note: The gskkyman utility changed in z/OS Version 1 Release 4. Use this example only if your system is at Version 1 Release 4 or a later release.

If you have set up your secure environment using either the gskkyman utility or the IKEYMAN utility, you can migrate the environment to a Resource Access Control Facility (RACF) database. RACF provides a more secure environment for your certificates and keys because they are stored in a database rather than in a

z/OS UNIX System Services file. This example includes steps to migrate your gskkyman key database or your IKEYMAN key database to a RACF key ring. You can use this example whether your server certificate was signed by an internal certificate authority (CA) or an external CA.

Note: No difference exists between certificates, keys, and databases that are created with IKEYMAN, and those that are created with gskkyman. Even though you created your server key database with IKEYMAN when you were on an earlier release of the Web server, you can manage that key database using gskkyman now that you have moved to a version of the Web server that uses System SSL. The example uses the gskkyman utility to migrate your certificates and keys to RACF.

System SSL notes:

You must use System SSL to establish secure connections. To use System SSL with the Web server, the System SSL load library must exist in linklist and must be under program control. If you have not already done so:

- Add the load library to the linklist.
- Turn on program control for the library by issuing the following RACF commands from a user ID that has the proper authority:

```
RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

  If turning on program control for the first time, use the RDEFINE command instead of the RALTER command.

RACF notes:

- The user ID issuing the RACF commands must have the proper authority.
- Choose the environment in which you execute the RACF commands (TSO READY, ISPF option 6, and so forth). Implementing these commands varies from one environment to another. See your local RACF administrator for assistance, or review the appropriate books for your environment. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Summary of tasks

In this example, you perform the following main tasks:

1. Create the server key ring in RACF.
2. Insure that your CA certificate exists in RACF and has a status of TRUST.
3. Insure that your server certificate and private key exist in RACF.
4. Permit the ID used to start the server to access the key ring.
5. Register the server key ring with the Web server.
6. Use server security defaults.
7. Verify that you can establish a secure connection with the server.
8. Set up client authentication as an option by using client certificates.

Step 1. Create the server key ring in RACF

A server key ring is required for each server that clients connect to using a secure SSL connection. Specify the key ring name on the KeyFile directive in the server configuration file, `httpd.conf`.

Assign server certificates and server key rings to the same user ID used to start the Web server. This action is required for SSL to initialize.To ensure you use the correct user ID, check the Web server header in the job log or vv trace, and find the line beginning with **Running as**. The Web server user ID appears in quotes. In the following example WEBSRV represents the user ID used to start the Web server:

```
........... This is IBM HTTP Server V5R3M0
........... Built on Sep  5 2000 at 09:58:01.
........... Started at Thu Sep  7 15:18:48 2000
........... Running as "WEBSRV", UID:0, GID:205.
```

In this step, you create:

- **SERVER1**, the server key ring.

To create the server key ring, **SERVER1**, issue the RACF command:

```
RACDCERT ID(WEBSRV) ADDRING(SERVER1)
```

Step 2. Insure that your CA certificate exists in RACF and has a status of TRUST

Your CA certificate must exist in the RACF list of CA certificates. The certificate must have a status of **TRUST** before you can receive the CA-signed certificate into your server key ring.

By default, the following CAs are designated in RACF, with a status of **NO TRUST**. Before you can use one of these CAs in the list, you must first mark that CA with a status of **TRUST**.

- Integrion Certification Authority Root
- IBM World Registry Certification Authority
- Thawte Personal Premium CA
- Thawte Personal Freemail CA
- Thawte Personal Basic CA
- Thawte Premium Server CA
- Thawte Server CA
- RSA Secure Server Certification Authority
- VeriSign Class 1 Public Primary Certification Authority
- VeriSign Class 2 Public Primary Certification Authority
- VeriSign Class 3 Public Primary Certification Authority

In this step, you do the following:

- Check whether your CA appears in the list of CAs in RACF and whether it has a status of **TRUST**.
- If the CA certificate appears in the list of CAs in RACF, but the status is **NO TRUST**, change the status to **TRUST**.
- Add the CA certificate to RACF and mark this certificate with a status of **TRUST** if it is not in the list of CAs in RACF.
- Connect the CA certificate to your **SERVER1** key ring.

To check the list of CAs, issue the following command:

```
RACDCERT CERTAUTH
```

If your CA is in the list of CAs in RACF, but has a current status of **NO TRUST**, then change the status to **TRUST**. For example, issue the following command:

```
RACDCERT CERTAUTH ALTER(LABEL('CA cert for server1')) TRUST
```

where:

- **CERTAUTH** indicates the type of certificate you are altering in RACF, in this case, a certificate authority certificate.
- **CA cert for server1** is the label for the CA certificate.

If your CA certificate is not in the list of CAs in RACF, you can obtain the CA certificate from your server key database that you created. You could have created the database with gskkyman or IKEYMAN. Do the following to copy the CA certificate into a file.

1. Change to the path where the server key database is located. In this example, the path is `/ibm/security/user1`.
2. Enter `gskkyman` to start the utility.
3. Enter **2** to open your server key database.
4. Enter the server key database name. In this example, that name is `server1.kdb`.
5. Enter the database password.
6. Enter **2** to manage your certificates.
7. Select the CA certificate that signed your server certificate.
8. Enter **4** to export the certificate to a file.
9. Indicate that you want to save the output certificate file as Base64 ASN.1 DER by entering **2**.
10. Enter the export file name. In this example, that name is `cert1.arm`
11. Press **Enter**, then enter **0** to exit the utility, after the gskkyman utility successfully exports your CA certificate.

Use the TSO/E **OGET** command to move the certificate from the hierarchical file system (HFS) into an MVS sequential data set. For example, issue the following command:

```
oget '/ibm/security/user1/cert1.arm' cert1.arm
```

Note: You can execute this TSO/E command from TSO/E, ISPF option 6, and the shell. To find out more about this command, including where to execute it, please see the *z/OS UNIX System Services Command Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

After receiving the CA certificate, add the certificate to RACF with **TRUST** status. For example, issue the following RACF command:

```
RACDCERT CERTAUTH ADD(CERT1.ARM) TRUST WITHLABEL('CA cert for server1')
```

where

- **CERTAUTH** indicates the type of certificate you are adding to RACF, in this case, a certificate authority certificate.
- **CERT1.ARM** is the data set containing the CA certificate.
- **CA cert for server1** is the label for the CA certificate.

Now that your CA certificate is in the RACF list of CA certificates and has a status of trust, connect the CA certificate to your **SERVER1** key ring. To connect the CA certificate to the key ring, issue the following RACF command:

```
RACDCERT ID(WEBSRV) CONNECT(CERTAUTH LABEL('CA cert for server1') RING(SE
RVER1)
```

where:

- **WEBSRV** is the ID used to start the Web server and the ID under which the server key ring resides.

- **CA cert for server1** is the label for the CA certificate.
- **CERTAUTH** indicates the type of certificate connected, in this case, a certificate authority certificate.

Step 3. Insure that your server certificate and private key exist in RACF

In this step, you move a copy of your server certificate and private key from your key database in the hierarchical file system (HFS) to a server key ring in RACF. You accomplish this by doing the following:

- Exporting your server certificate and private key from your key database to a file in the HFS
- Moving the server certificate and private key from the HFS into an MVS sequential data set
- Adding the server certificate to RACF with **TRUST** status
- Connecting the server certificate to your **SERVER1** key ring and making this certificate the default in the key ring

Export your server certificate and private key from your key database to a file in the HFS, by doing the following:

1. Change to the path where the server key database is located. In this example, the path is `/ibm/security/user1`.
2. Enter `gskkyman` to start the utility.
3. Enter **2** to open your server key database.
4. Enter the server key database name. In this example, that name is `server1.kdb`.
5. Enter the database password.
6. Enter **1** to manage keys and certificates.
7. Select the label of the signed server certificate in the list.
8. Enter **7** to export the certificate and key to a file.
9. Specify the export file format. For this example specify one of the binary options.
10. Enter the name of the PKCS12 formatted file that contains the signed server certificate, for example, `server1.p12`.
11. Enter a password using the guidelines in "Key database password" on page 158. You need this password to add your server certificate to RACF.
12. Enter the password again for verification.
13. Enter **1** for strong encryption, or **0** export encryption.
14. Press Enter, then enter **0** to exit the utility, after the gskkyman utility successfully exports your server certificate and server key.

Use the TSO/E **OGET** command to move the server certificate and private key from the HFS into an MVS sequential data set. Since the server certificate is in PKCS12 format, store the certificate on MVS in variable block (VB) format. If you use the **OGET** command, you insure that the data set is in variable block format. If you move the certificate into a new data set, the **OGET** command creates a VB sequential data set by default.

```
oget '/ibm/security/user1/server1.p12' server1.p12 binary
```

Note: You can execute this TSO/E command from TSO/E, ISPF option 6, and the shell. To find out more about this command, including where to execute it, please see the *z/OS UNIX System Services Command Reference*. To access this

book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Add the server certificate to RACF with **TRUST** status. The private key is also added. For instance, issue the following RACF command:

```
RACDCERT ID(websrv) ADD(SERVER1.P12) WITHLABEL('Certificate for server1')
PASSWORD('X4RB')
```

where:

- **websrv** is the ID used to start the Web server.
- **SERVER1.P12** is the data set containing the server certificate.
- **Certificate for server1** is the label for the server certificate.
- **X4RB** is the required password for PKCS12 certificates. This password is case-sensitive.

Connect the server certificate that is now in RACF to your **SERVER1** key ring and make the certificate the default certificate in the key ring. For example, issue the following RACF command:

```
RACDCERT ID(WEBSRV) CONNECT(ID(WEBSRV) LABEL('Certificate for server1')
RING(SERVER1) DEFAULT)
```

where:

- **WEBSRV** is the ID used to start the Web server and the ID under which the server key ring and the server certificate reside.
- **SERVER1** is the server key ring.
- **Certificate for server1** is the label that identifies the key and certificate in the key ring.
- **DEFAULT** makes the server certificate the default in the key ring.

Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.

Step 4. Permit the ID used to start the server to access the key ring

In this step you permit the user ID **WEBSRV** to access the key ring through the **DIGTCERT** general resource class.

The ID used to start the Web server must have access to the key ring created using RACDCERT. If the ID does not have access, SSL initialization fails. In this example the Web server startup ID is WEBSRV. To permit WEBSRV to access the server key ring, you issue RACF commands to perform the following tasks:

- Define the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING resources with universal access of `None`.
- Permit the WEBSRV ID read access to the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING resources in the FACILITY class.
- Activate the FACILITY general resource class.
- Refresh the FACILITY general resource class.

To perform these tasks, issue the following commands:

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
PE IRR.DIGTCERT.LIST CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
```

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PE IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)

SETR CLASSACT(FACILITY)
SETR RACLIST(FACILITY) REFRESH
```

To find out more about controlling access to the RACDCERT function through the FACILITY general resource class, see the *z/OS Security Server (RACF): Security Administrator's Guide* and the description of the RACDCERT command in the *z/OS Security Server (RACF): Security Administrator's Guide*. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Step 5. Register the server key ring with the Web server

To register your server key ring in this example with the Web server, add the following KeyFile directive to your server configuration file, `httpd.conf`:

```
KeyFile SERVER1 SAF
```

Step 6. Use server security defaults

The security default settings for this example follow:

SSLMode
: Support for SSL connections is set **on**.

SSLPort
: Port 443 is the default port used for SSL.

SSLCipherSpec
: The server uses an encryption level, supported by both the client and the server.

You can change the Web server security defaults, by editing the appropriate security directives in the configuration file.

Step 7. Verify that you can establish a secure connection with the server

To verify that you can establish a secure connection with your server, start your Web server. Then start a browser and enter the following URL:

```
https://your_server_name:SSL_port_number
```

The URL for this example is:

```
https://serva.raleigh.ibm.com:443
```

Step 8. Set up client authentication as an option by using client certificates

For instructions, see "Example 8 for RACDCERT: Setting up client authentication using client certificates signed by an internal CA" on page 135, or "Example 9 for RACDCERT: Setting up client authentication using client certificates signed by an external CA" on page 140.

Example 10: Migrating your secure environment from either a gskkyman key database or an IKEYMAN key database to a RACF key ring (z/OS Version 1 Release 3 or earlier)

Note: The gskkyman utility changed in z/OS Version 1 Release 4. Use this example only if your system is at Version 1 Release 3 or an earlier release.

If you have set up your secure environment using either gskkyman or IKEYMAN, you can migrate the environment to a Resource Access Control Facility (RACF) database. RACF provides a more secure environment for your certificates and keys because they are stored in a database rather than in a z/OS UNIX System Services file. This example includes steps to migrate your gskkyman key database or your IKEYMAN key database to a RACF key ring. You can use this example whether your server certificate was signed by an internal certificate authority (CA) or an external CA.

Note: No difference exists between certificates, keys, and databases that are created with IKEYMAN, and those created with gskkyman. Even though you created your server key database with IKEYMAN when you were on an earlier release of the Web server, you can manage that key database using gskkyman now that you have moved to a version of the Web server that uses System SSL. The example uses gskkyman to migrate your certificates and keys to RACF.

System SSL notes:

You must use System SSL to establish secure connections. To use System SSL with the Web server, the System SSL load library must exist in linklist and must be under program control. If you have not already done so:

- Add the load library to the linklist.
- Turn on program control for the library by issuing the following RACF commands from a user ID that has the proper authority:

```
RALTER PROGRAM * ADDMEM('hlq.SGSKLOAD'//NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

  If turning on program control for the first time, use the RDEFINE command instead of the RALTER command.

RACF notes:

- The user ID issuing the RACF commands must have the proper authority.
- Choose the environment in which you execute the RACF commands (TSO READY, ISPF option 6, and so forth). Implementing these commands varies from one environment to another. See your local RACF administrator for assistance, or review the appropriate books for your environment. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

For an explanation of terms, refer to "Terms used in these examples" on page 73. For additional information on security concepts, terminology, and configuration options, refer to the books and resources in "Related documentation and URLs" on page 159.

Summary of tasks

In this example, you perform the following main tasks:

1. Create the server key ring in RACF.
2. Insure that your CA certificate exists in RACF and has a status of TRUST.
3. Insure that your server certificate and private key exist in RACF.
4. Permit the ID used to start the server to access the key ring.
5. Register the server key ring with the Web server.
6. Use server security defaults.
7. Verify that you can establish a secure connection with the server.
8. Set up client authentication as an option by using client certificates.

Step 1. Create the server key ring in RACF

A server key ring is required for each server that clients connect to using a secure SSL connection. Specify the key ring name on the KeyFile directive in the server configuration file, `httpd.conf`.

Assign server certificates and server key rings to the same user ID used to start the Web server. This action is required for SSL to initialize.To ensure you use the correct user ID, check the Web server header in the job log or vv trace, and find the line beginning with **Running as**. The Web server user ID appears in quotes. In the following example WEBSRV represents the user ID used to start the Web server:

```
........... This is IBM HTTP Server V5R3M0
........... Built on Sep  5 2000 at 09:58:01.
........... Started at Thu Sep  7 15:18:48 2000
........... Running as "WEBSRV", UID:0, GID:205.
```

In this step, you create:

- **SERVER1**, the server key ring.

To create the server key ring, **SERVER1**, issue the RACF command:

```
RACDCERT ID(WEBSRV) ADDRING(SERVER1)
```

Step 2. Insure that your CA certificate exists in RACF and has a status of TRUST

Your CA certificate must exist in the RACF list of CA certificates. The certificate must have a status of **TRUST** before you can receive the CA-signed certificate into your server key ring.

By default, the following CAs are designated in RACF, with a status of **NO TRUST**. Before you can use one of these CAs in the list, you must first mark that CA with a status of **TRUST**.

- Integrion Certification Authority Root
- IBM World Registry Certification Authority
- Thawte Personal Premium CA
- Thawte Personal Freemail CA
- Thawte Personal Basic CA
- Thawte Premium Server CA
- Thawte Server CA
- RSA Secure Server Certification Authority
- VeriSign Class 1 Public Primary Certification Authority
- VeriSign Class 2 Public Primary Certification Authority
- VeriSign Class 3 Public Primary Certification Authority

In this step, you do the following:

- Check whether your CA appears in the list of CAs in RACF and whether it has a status of **TRUST**.
- If the CA certificate appears in the list of CAs in RACF, but the status is **NO TRUST**,change the status to **TRUST**.
- Add the CA certificate to RACF and mark this certificate with a status of **TRUST** if it is not in the list of CAs in RACF.
- Connect the CA certificate to your **SERVER1** key ring.

To check the list of CAs, issue the following command:

```
RACDCERT CERTAUTH
```

If your CA is in the list of CAs in RACF, but has a current status of **NO TRUST**, then change the status to **TRUST**. For example, issue the following command:

```
RACDCERT CERTAUTH ALTER(LABEL('CA cert for server1')) TRUST
```

where:

- **CERTAUTH** indicates the type of certificate you are altering in RACF, in this case, a certificate authority certificate.
- **CA cert for server1** is the label for the CA certificate.

If your CA certificate is not in the list of CAs in RACF, you can obtain the CA certificate from your server key database that you created. You could have created the database with gskkyman or IKEYMAN. Do the following to copy the CA certificate into a file.

1. Change to the path where the server key database is located. In this example, the path is `/ibm/security/user1`.
2. Enter **gskkyman** to start the utility.
3. Enter **2** to open your server key database.
4. Enter the server key database name. In this example, that name is `server1.kdb`.
5. Enter the database password.
6. Enter **10** to list all trusted CAs.
7. Enter the key number for the CA certificate that signed your server certificate.
8. Enter **5** to copy the certificate to a file.
9. Enter **1** to save the file in Base 64-encoded ASCII format.
10. Enter the certificate file name. In this example, that name is `cert1.arm`
11. Enter **1** to exit the gskkyman utility, after receiving a message that your request completed successfully.

Use the TSO/E OGET command to move the certificate from the hierarchical file system (HFS) into an MVS sequential data set. For example, issue the following command:

```
oget '/ibm/security/user1/cert1.arm' cert1.arm
```

Note: You can execute this TSO/E command from TSO/E, ISPF option 6, and the shell. To find out more about this command, including where to execute it, please see the *z/OS UNIX System Services Command Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

After receiving the CA certificate, add the certificate to RACF with **TRUST** status. For example, issue the following RACF command:

```
RACDCERT CERTAUTH ADD(CERT1.ARM) TRUST WITHLABEL('CA cert for server1')
```

where

- **CERTAUTH** indicates the type of certificate you are adding to RACF, in this case, a certificate authority certificate.
- **CERT1.ARM** is the data set containing the CA certificate.
- **CA cert for server1** is the label for the CA certificate.

Now that your CA certificate is in the RACF list of CA certificates and has a status of trust, connect the CA certificate to your **SERVER1** key ring. To connect the CA certificate to the key ring, issue the following RACF command:

```
RACDCERT ID(WEBSRV) CONNECT(CERTAUTH LABEL('CA cert for server1') RING(SE
RVER1)
```

where:

- **WEBSRV** is the ID used to start the Web server and the ID under which the server key ring resides.
- **CA cert for server1** is the label for the CA certificate.
- **CERTAUTH** indicates the type of certificate connected, in this case, a certificate authority certificate.

Step 3. Insure that your server certificate and private key exist in RACF

In this step, you move a copy of your server certificate and private key from your key database in the hierarchical file system (HFS) to a server key ring in RACF. You accomplish this by doing the following:

- Exporting your server certificate and private key from your key data base to a file in the HFS
- Moving the server certificate and private key from the HFS into an MVS sequential data set
- Adding the server certificate to RACF with **TRUST** status
- Connecting the server certificate to your **SERVER1** key ring and making this certificate the default in the key ring

Export your server certificate and private key from your key database to a file in the HFS, by doing the following:

1. Change to the path where the server key database is located. In this example, the path is `/ibm/security/user1`.
2. Enter **gskkyman** to start the utility.
3. Enter **2** to open your server key database.
4. Enter the server key database name. In this example, that name is `server1.kdb`.
5. Enter the database password.
6. Enter **9** to export the certificate. (export keys)
7. Enter **2** to export the certificate to a PKCS12 file.
8. Select the label of the signed server certificate in the list.
9. Enter the name of the PKCS12 formatted file that will contain the signed server certificate, for example, `server1.p12`.
10. Enter a password using the guidelines in “Key database password” on page 158. You need this password to add your server certificate to RACF.
11. Enter the password again for verification.
12. Enter **1** to exit gskkyman.

Use the TSO/E OGET command to move the server certificate and private key from the HFS into an MVS sequential data set. Since the server certificate is in PKCS12 format, you must store the certificate on MVS in variable block (VB) format. If you use the OGET command, you insure that the data set is in variable block format. If you are moving the certificate into a new data set, OGET creates a VB sequential data set by default.

```
oget '/ibm/security/user1/server1.p12' server1.p12 binary
```

Note: You can execute this TSO/E command from TSO/E, ISPF option 6, and the shell. To find out more about this command, including where to execute it, please see the *z/OS UNIX System Services Command Reference*. To access this

book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Add the server certificate to RACF with **TRUST** status. The private key is also added. For instance, issue the following RACF command:

```
RACDCERT ID(websrv) ADD(SERVER1.P12) WITHLABEL('Certificate for server1')
PASSWORD('X4RB')
```

where:

- **websrv** is the ID used to start the Web server.
- **SERVER1.P12** is the data set containing the server certificate.
- **Certificate for server1** is the label for the server certificate.
- **X4RB** is the password, which is required for PKCS12 certificates. This password is case-sensitive.

Connect the server certificate that is now in RACF to your **SERVER1** key ring and make the certificate the default certificate in the key ring. For example, issue the following RACF command:

```
RACDCERT ID(WEBSRV) CONNECT(ID(WEBSRV) LABEL('Certificate for server1')
RING(SERVER1) DEFAULT)
```

where:

- **WEBSRV** is the ID used to start the Web server and the ID under which the server key ring and the server certificate reside.
- **SERVER1** is the server key ring.
- **Certificate for server1** is the label that identifies the key and certificate in the key ring.
- **DEFAULT** makes the server certificate the default in the key ring.

Note: Alternatively, you can use the SSLServerCert directive to select a server certificate for a particular server IP address. For more information, see "SSLServerCert - Associate a key database with an IP address" on page 551.

Step 4. Permit the ID used to start the server to access the key ring

In this step you permit the user ID **WEBSRV** to access the key ring through the **DIGTCERT** general resource class.

The ID used to start the Web server must have access to the key ring created using RACDCERT. If the ID does not have access, SSL initialization fails. In this example the Web server startup ID is WEBSRV. To permit WEBSRV to access the server key ring, you issue RACF commands to perform the following tasks:

- Define the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING resources with universal access of `None`.
- Permit the WEBSRV ID read access to the IRR.DIGTCERT.LIST and IRR.DIGTCERT.LISTRING resources in the FACILITY class.
- Activate the FACILITY general resource class.
- Refresh the FACILITY general resource class.

To perform these tasks, issue the following commands:

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
PE IRR.DIGTCERT.LIST CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
```

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PE IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)

SETR CLASSACT(FACILITY)
SETR RACLIST(FACILITY) REFRESH
```

To find out more about controlling access to the RACDCERT function through the FACILITY general resource class, see the *z/OS Security Server (RACF): Security Administrator's Guide* and the description of the RACDCERT command in the *z/OS Security Server (RACF): Security Administrator's Guide*. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Step 5. Register the server key ring with the Web server

To register your server key ring in this example with the Web server, add the following KeyFile directive to your server configuration file, `httpd.conf`:

```
KeyFile SERVER1 SAF
```

Step 6. Use server security defaults

The security default settings for this example follow:

SSLMode
: Support for SSL connections is set **on**.

SSLPort
: Port 443 is the default port used for SSL.

SSLCipherSpec
: The server uses an encryption level, supported by both the client and the server.

You can change the Web server security defaults, by editing the appropriate security directives in the configuration file.

Step 7. Verify that you can establish a secure connection with the server

To verify that you can establish a secure connection with your server, start your Web server. Then start a browser and enter the following URL:

```
https://your_server_name:SSL_port_number
```

The URL for this example is:

```
https://serva.raleigh.ibm.com:443
```

Step 8. Set up client authentication as an option by using client certificates

For instructions, see "Example 8 for RACDCERT: Setting up client authentication using client certificates signed by an internal CA" on page 135, or "Example 9 for RACDCERT: Setting up client authentication using client certificates signed by an external CA" on page 140.

Related information

Certificate file types generated by the gskkyman utility

z/OS Version 1 Release 4 or later

When you export your self-signed CA certificate with the gskkyman utility, you have the option to export the certificate in the following formats:

- Binary ASN.1 DER

- Base64 ASN.1 DER
- Binary PKCS#7
- Base64 PKCS#7

If you export the file in Base 64 ASN.1 DER format, use a file extension of `.arm`. If you export the file in Binary ASN.1 DER format, use a file extension of `.der`. Some major browsers do not accept the PKCS#7 format. Therefore, only use the base64 ASN.1 DER format or the binary ASN.1 DER format.

z/OS Version 1 Release 3 or earlier

The following certificate file types are generated by gskkyman:

- ASCII file types .arm and .cer (base64-encoded format)
- Binary file type .crt (DER-encoded format)

When you create a self-signed server or CA certificate with gskkyman, you are given the option to press enter to use the default file name and type (cert.arm) or specify a file name and type. When you create a certificate request with gskkyman, you are given the option to select an ASCII or binary file type. If you select **1** (ASCII), the default file created is certreq.arm; if **2** (binary), the default file created is cert.crt.

Notes:

1. When you receive a certificate using the Netscape or Microsoft Internet Explorer browser, most newer versions will automatically start a wizard to step you through the process. For certain versions of Microsoft Internet Explorer, you may need to select the option to open the file rather than saving it to disk to start the wizard.
2. If your server and client certificates will be signed by an external commercial CA, such as VeriSign or Thawte, check to see which file type they request before generating your certificate request.

Key database password

When you create a new key database, you specify a key database password. This password is important because it protects the private key. The private key is the only key that can sign documents or decrypt messages encrypted with the public key. It is good practice to change the key database password frequently.

Use the following guidelines when specifying the password:

- The password must be from the U.S. English character set.
- The password should be at least six characters and contain at least two nonconsecutive numbers. Make sure the password doesn't consist of publicly obtainable information about you, such as the initials and birth date for you, your spouse, or children.

Note: If you specify an expiration date for the password, keep track of when to change it. If the password expires before you change it, a message will be written to the error log. The server will start, but there will not be a secure network connection if the password has expired.

asn.1 encoding and decoding error

When you go through the steps for setting up a secure Web server environment using SSL, you can receive an error message similar to the following:

```
An asn.1 encoding/decoding error occurred.
```

You typically encounter the error when you process certificates with a utility. However, you can also receive this error when you are processing other things such as PKCS12 files and key pairs. The error typically indicates that something is wrong with the file. However, check APARs or code fixes to make sure that no error exists with the utility. You can encounter the error with gskkyman, or with another utility such as a browser utility that you use to read a certificate into a browser.

There is some sort of encoding error in your file, which can be introduced in a variety of ways:

- Trailing blanks can accidentally be introduced through file editing or improper file transfer.
- Trailing line numbers can be introduced through editing.
- Other characters can be in introduced through file editing or improper file transfer.

Transfer your certificates and certificate requests between your workstation and MVS by using the ASCII option of File Transfer Protocol (FTP) or an equivalent file transfer program. This allows ASCII to EBCDIC conversion, and treats the file as character-encoded, not binary. Transfer `.kdb` files and `.rdb` files in binary format.

Resolve the the problem by doing the following:

- Review the files and correct any errors that you find.
- If you cannot find the error, review the security examples in this chapter, "Examples: Setting up secure connections" on page 69. Correct any error that you find in the steps that you executed.
- If you still cannot find the error, redo the steps for setting up your secure environment. Carefully follow the steps of the examples in "Examples: Setting up secure connections" on page 69.

Related documentation and URLs

z/OS documentation

The following resources provide additional information on setting up and using System SSL and RACF:

- *z/OS System SSL Programming Guide and Reference*. This book provides more detailed information on setting up and using the z/OS System SSL gskkyman utility. It also shows you how to perform gskkyman tasks using the RACF RACDCERT command. To access this book on the Web, go to URL http://www.ibm.com/servers/eserver/zseries/zos/bkserv/.
- *z/OS Security Server (RACF): Security Administrator's Guide*. To access this guide and other RACF books on the Web, go to URL http://www.ibm.com/servers/eserver/zseries/zos/bkserv/.

Security documentation

For more information on security and links to other resources, go to the IBM Security Library Web site at URL http://www.ibm.com/security/library/.

Setting up SSL support for multiple IP addresses

To configure your server for SSL support of multiple IP addresses, edit the server configuration file directly or use the Configuration and Administration Forms:

- If you edit the configuration file directly, use the SSLServerCert directive. For a description of this directive, see "SSLServerCert - Associate a key database with an IP address" on page 551.
- To use the Configuration and Administration Forms:
  1. From the Front Page of your server, click **CONFIGURATION AND ADMINISTRATION FORMS**.
  2. Click **Security Configuration**, then click **Key Database Configuration**.
  3. Scroll down to **Multiple key entries** to enter the required information.

Related information:

- For a description of this option, see "SSL support for multiple IP addresses" on page 62.

Changing the default order of encryption levels used by the Web server

To specify the order of encryption levels used by the Web server, edit the server configuration file directly or use the Configuration and Administration Forms:

- If you edit the configuration file directly, update the SSLCipherSpec directive. For a description of this directive, see "SSLCipherSpec - Specify the levels of encryption to use for SSL connections" on page 547.
- To use the Configuration and Administration Forms:
  1. From the Front Page of your server, click **CONFIGURATION AND ADMINISTRATION FORMS**.
  2. Click **Security Configuration**, then click **SSL Security Options**.
  3. Scroll down to **Specify SSL cipher specifications** to select the encryption settings for your server.

Chapter 9. Setting up protection for server resources

Before you begin 161
Protection methods 161
Step 1. Activate protection on the server 162
Step 2. Specify which requests you want the server to accept 162
Step 3. Decide which protection options to use . . 163
Options for protecting server resources 163
User names and password protection . . . 163
Address template protection 164
SSL client authentication. 164
How the server processes requests 164
Step 4. Create protection setups 165
Directives 165
Subdirectives 166
ServerID — Identifying the protection setup to requesters. 166
AuthType — Specifying the type of authentication 166
PasswdFile — Pointing to the password file 167
GroupFile — Pointing to a server group file 167
Mask — Specifying valid user names, groups, and addresses 167
Creating protection setups for Secure Sockets Layer (SSL) client authentication 168
Hints and tips for coding SSL client authentication parameters for protection setups 170
Step 5. Limit access to individual files 170
Rules for specifying user names, group names, and address templates 171
Group files 172
Use of group files in protection setups 173
Access Control List files 174
Examples. 175
Secure Sockets Layer client authentication hints and tips 176
Default protection scheme when coding %%CLIENT%% on UserID directive 176
Examples: Protecting server resources 177
Example 1: Setting up protection with user names and passwords 177
Step 1. Accept the Web server default user ID %%CLIENT%%. 177
Step 2. Create a URL request template for the resource that needs to be protected and specify protection settings for that request . . 177
Step 3. Set up mapping rules to route the request to the correct resource 178
Step 4. Restart the server to use your new configuration settings. 178
Example 2: Setting up a protection setup without SSL client authentication 178
Example 3: Setting up a protection setup with SSL client authentication. 181
Changing expired passwords 182

Before you begin

This chapter describes how to protect the data and files on your server. For information on setting up security and secure connections, see Chapter 8, “Setting up a secure server” on page 53.

Protection methods

You announce to the world that you want people to come look at the information on your server. But once you publicize your server to the Internet, you risk attracting unwanted attention to the system on which it runs. Unauthorized people may try to guess passwords, update or delete files, execute programs, or access confidential data. Part of the attraction of the World Wide Web is its openness. However, the Web is open to both positive use and abuse.

There are several ways that you can protect your system:

- Place a server meant for public access in a network that is separate from your local or internal network. A firewall provides a way to separate an internal network from a publicly accessible network, such as the Internet. The firewall can be a group of computers or a single computer that acts as a gateway in both directions, regulating and tracking the traffic passing through it. You can configure your server as a proxy server through a firewall. This way, internal users can get out to the Internet, but outside Internet travelers cannot get in. Using a proxy server also ensures that any access to the Internet by internal

browsers is done anonymously. Because all of the requests to the Internet are made by the proxy server on the part of the internal user, the remote host does not have access to the name of the specific host that is on the inside of the firewall.

- Disable Telnet, rlogin, and finger clients on the system that is running the server. In particular, consider disabling Telnet and TN3270.
- Use packet filtering and firewalls. Packet filtering allows you to define where data can come from and where it can go. You can configure your system to reject certain source/destination combinations.
- Protect access to CGI scripts. Using CGI scripts on a Web server can create a security exposure. If you use CGI scripts, you need to protect them by controlling who has access to them. It is possible to write CGI scripts that display all environment variables. At times these variables may include sensitive data such as user IDs and passwords. So you must be careful about displaying environment variables in your CGI scripts and control access to your CGI scripts. Make sure you know what a CGI program does before you make it available on your server.
- Protect user directories on your server. By allowing local users to have Web pages, you may be creating a security exposure. You need to carefully control access to the pages on local users' machines. You may choose not to allow any local users to have executable CGI scripts. For more information, see "User directories" on page 469.

Step 1. Activate protection on the server

The first step to controlling access to your server's resources is to activate protection. You activate protection based on the content of requests that clients send to your server. A ***request*** is the part of a URL that follows your server host name. For example, the following URL is entered by a requester:

```
http://myserver.raleigh.ibm.com/test/schedule.html
```

In this URL, `myserver.raleigh.ibm.com` is the server host name, and `test/schedule.html` is the request.

You can use Protect directives to specify which requests should activate protection. Each Protect directive has a request template. The server activates protection when it receives a request that matches a request template on a Protect directive. The Protect directive also either identifies or contains the protection setup to be used.

You also need to control access to your server's resources by establishing the access control environment. You should specify the local MVS user ID to use when processing the request on the Protect or DefProt directive.

See "Access control - Set up access control for the server" on page 435 for details on how to use Protect directives.

Step 2. Specify which requests you want the server to accept

Besides activating protection for the requests, you must also tell your server which requests to accept for processing.

Use Pass and Exec directives to specify which requests you want your server to accept. The Pass and Exec directives map requests to actual directories and files on your server (the information you are protecting).

Like the Protect directive, each Pass and Exec directive contains a request template. After the server checks to see if a request activates protection, it goes through its list of Pass and Exec directives to determine if it should accept the request. If the request matches a request template on a Pass or Exec directive, the server ***accepts*** the request. If protection was activated, the server uses the protection setup to determine whether it should ***complete*** the request.

So for protection to work properly, you must ensure that your Pass and Exec directives accept the requests that your Protect directives activate protection for. Use Pass directives to accept document requests. Use Exec directives to accept CGI program requests.

Note:
Protect directives must be placed before any Pass or Exec directives in your configuration file.

See "Resource mapping - Redirect URLs" on page 536 for details on how to use the Pass and Exec directives.

Step 3. Decide which protection options to use

Options for protecting server resources

There are three protection options you can use to control access to the resources on your server:

- "User names and password protection"
- "Address template protection" on page 164
- "SSL client authentication" on page 164

You can use one protection option or a combination of options.

User names and password protection

With this type of protection, you create user names for the requesters who are authorized to access your protected resources. When you define each user name, you assign a password for that user. You can assign the same password to multiple users. You can also set up user group files.

After you define a user name, you can use it within protection setups and ACL files to specify which user names are valid for different types of requests.

When the server receives a request that activates this type of protection, the server prompts the requester for a user name and password. In order for the request to complete, the requester must return a user name and password that meet the following criteria:

- The user name must be defined in the protection setup or ACL file as valid for the type of request being made.
- The password must match the one defined for the user name in the password file.
- %%SAF%% can be used in place of a password file for this protection setup. If you use this method, a System Authorization Facility (SAF)-based security product, such as z/OS RACF, can be used to authenticate the requester using SAF-based user IDs and passwords.

Address template protection

With this type of protection, you use address templates to specify valid requester addresses for the different types of requests. You can use address templates in protection setups and ACL files.

When the server receives a request that activates this type of protection, it compares the address of the requester to the templates to determine if the request comes from a valid address. The server can use either the IP address of the requester (for example, 9.67.97.103) or the host name of the requester (for example, myserver.raleigh.ibm.com) when comparing against the templates.

Note: In order to compare the requester host names against address templates, you must set the DNS-Lookup directive to `On`. If the DNS-Lookup directive is set to `Off` (the default), your server can compare only the IP address of the requester to the address templates. See "DNS-Lookup - Specify whether you want to look up host names of clients" on page 450.

SSL client authentication

SSL Version 3 is required for client authentication. If you set up your server for SSL client authentication, the server will request a certificate from clients making an **https** request. You can configure the server to require a certificate for all Web pages or selected pages.

The server will establish a secure connection if the client has a valid certificate. The server will deny the request if the client has a certificate that has expired or if the certificate is signed by a CA who is not designated as a trusted CA on the server.

You can optionally map a client certificate to an existing z/OS RACF user ID and password so that RACF authenticates the client.

How the server processes requests

The following steps describe how the server processes a request that has activated protection and been accepted by a Pass or Exec directive. These steps assume that all protection is defined in the protection setup (no ACL file exists on the protected directory):

1. Based on the HTTP method of the request, the server refers to the appropriate mask subdirective (DeleteMask, GetMask, Mask, PostMask, or PutMask) in the protection setup. The mask subdirective specifies valid user names, groups, or address templates.
2. If any items on the mask subdirective use only address template protection, the server compares the address of the requester against the address templates.

   Items that use only address template protection start with either `@`, `Anybody@`, `Anyone@`, or `Anonymous@`, followed by one or more address templates. Group names on the subdirective might also contain items that use only address template protection.

   If there is a match, the server completes the request without prompting for a user name or password.

   If there is not a match or no items use address template protection only, the process continues with the next step.
3. If any items on the mask subdirective are user names or group names, the server prompts the requester for a user name and password.

4. The server checks the user name sent by the requester against the valid user names. Valid user names are either the individual user names on the mask subdirective or user names defined as being part of a group that is on the mask subdirective.

   If there is a match, the process continues with the next step.

   If there is not a match, the process ends and the server returns a message to the requester saying that authorization failed.
5. If the user name sent by the requester is also associated with an address template, the server checks the address of the requester against the template. The mask subdirectives and group files use the at sign character (@) to associate user names or group names with address templates.

   If there is a match, the process continues with the next step.

   If there is not a match, the process ends and the server returns a message to the requester saying that authorization failed.
6. The server checks the user name sent by the requester against the user names in the password file that the protection setup points to.

   If there is a match, the process continues with the next step.

   Note: It is important to note that the password file must contain an entry for the user name that the requester sends to the server. You make up the user names that are in the password file. The names themselves do not have any relation to the addresses of the requesters.

   If there is not a match, the process ends and the server returns a message to the requester saying that authorization failed.
7. The server checks the password sent by the requester against the password defined for the user name in the password file. Each user name in the password file has one valid password.

   If there is a match, the server completes the request.

   If there is not a match, the process ends and the server returns a message to the requester saying that authorization failed.

Step 4. Create protection setups

When a Protect directive activates protection for a request, it also either identifies the protection setup to use or defines the protection setup as part of the directive. A ***protection setup*** is a group of protection subdirectives. The subdirectives work together to define how the server should control access to the resources being protected.

Directives

You can create protection setups in the following ways:

Create protection setups within the configuration file as part of Protection, Protect, or DefProt directives

When you create a protection setup on a Protection directive, you give the setup a label that you can point to later from Protect and DefProt directives.

When you create a protection setup on a DefProt or Protect directive, the protection setup is used only for that directive. The setup cannot be pointed to by other DefProt or Protect directives. This type of protection setup is called an ***in-line*** protection setup.

You indicate you are including a protection setup as part of a Protection, Protect, or DefProt directive by making the last character on the line that contains the directive a left brace character ({). On each following line you put one protection subdirective and its value. You indicate the end of the protection setup by putting a right brace character (}) by itself on the line following the last protection subdirective. You cannot use comments within the protection setup.

See "Access control - Set up access control for the server" on page 435 for more information and examples of creating protection setups in the configuration file.

Create separate protection setup files

If you create separate protection setup files, you can point to those files from Protect and DefProt directives. A protection setup file is a plain text file. Within the file, each line contains one protection subdirective and the value for that subdirective.

Subdirectives

Within the protection setup, the protection subdirectives control access to the directory or files being protected. The following sections describe how to use each of the protection subdirectives.

ServerID — Identifying the protection setup to requesters

For user name and password protection, use the ServerID subdirective to specify a name you want to use to identify the protection setup to requesters. The name does not need to be a real machine name.

When the server sends a requester a prompt for user name and password, it also includes the name you specify on ServerID. Most browsers display this name with the prompt. Because different protection setups can use different password files, having a name associated with the protection setup can help the requester decide which user name and password to send back. Many browsers also attempt to automatically send a user name and password if the requester has previously responded to a prompt from a protection setup with the same name.

Because some browsers, such as NetScape, cache userid/password by security realm (ServerId) within host, we suggest you follow these guidelines when specifying ServerId and password files in your protection setups:

- Protection setups that use the same password file should use the same ServerId.
- Protection setups that use different password files should use different ServerIds.

If the protection setup is using address template protection only, you do not need to use the ServerID subdirective.

Example:

```
ServerID restricted
```

AuthType — Specifying the type of authentication

The AuthType subdirective specifies the type of authentication to use when a client sends a password to the server. For user name and password protection, you must use the AuthType subdirective with a value of `Basic`. With basic authentication, passwords are sent to the server as plain text. They are encoded, but not encrypted.

If the protection setup is using address template protection only, you do not need to use the AuthType subdirective.

Example:

```
AuthType Basic
```

PasswdFile — Pointing to the password file

For user name and password protection, use the PasswdFile subdirective to specify the path and name of the password file that you want the protection setup to use. You can also specify %%SAF%% to use a SAF-based security system, such as z/OS RACF, to perform the validation using your MVS system IDs and passwords.

If you use a password file, each password file contains a list of user names and passwords, and each user name has one valid password defined for it. The requester must send back a user name and password that exactly matches a user name and password in the password file. You create and maintain password files using the htadm command. See "htadm command" on page 389 for information on how to create and maintain password files.

Note: The user names in the password file do not have any relation to the addresses of the requesters. You specify user names and passwords of your choice.

Example: To use the SAF interface:

```
PasswdFile %%SAF%%
```

To use a password file:

```
PasswdFile /usr/lpp/internet/server_root/Admin/heroes.pwd
```

GroupFile — Pointing to a server group file

If you want to use group names in the protection setup, use the GroupFile subdirective to specify the path and file name of the server group file that contains the group definitions you want to use. The groups defined within the server group file can then be used by:

- Any mask subdirectives that are part of the protection setup. (The mask subdirectives are DeleteMask, GetMask, Mask, PostMask, and PutMask.)
- Any ACL file on a directory that is protected by the protection setup.

See "Group files" on page 172 more information about server group files.

Example:

```
GroupFile /docs/etc/WWW/restrict.group
```

Mask — Specifying valid user names, groups, and addresses

Use the mask subdirectives to specify valid user names, groups, and address templates for different types of requests. The mask subdirectives protect the entire directory that the request is mapped to.

Each request to your server contains an HTTP method field that identifies the type of request being made. See "Methods - Set method acceptance" on page 518 for a description of the HTTP methods supported by the server. Choose which mask subdirectives to use based on the types of requests you want to authorize. For a protection setup to be valid, it must contain at least one of the following mask subdirectives:

- DeleteMask - to authorize DELETE requests.

- GetMask - to authorize GET requests.
- PostMask - to authorize POST requests. (Most HTML forms use the POST method.)
- PutMask - to authorize PUT requests.
- Mask - to authorize requests using any enabled methods not covered by the other mask subdirectives. Other mask subdirectives take precedence over the Mask subdirective if both are present in the protection setup. For example, if a protection setup contains a DeleteMask subdirective and a Mask subdirective, DELETE requests are covered by the DeleteMask subdirective and all other requests are covered by the Mask subdirective.

For an explanation of how to specify user names, group names, and address templates, see "Rules for specifying user names, group names, and address templates" on page 171.

Creating protection setups for Secure Sockets Layer (SSL) client authentication

Chapter 8, "Setting up a secure server" on page 53 describes how to use Secure Sockets Layer (SSL) with a secure server. If you set up your server for SSL client authentication, the server requests a certificate from any client making an **https** request. The server establishes a secure connection whether or not the client has a valid certificate.

You can restrict document access by using password files, user authentication, and group authentication in protection setups as described in Chapter 9, "Setting up protection for server resources" on page 161. There are several additional ways you can restrict document access by using Distinguished Name information contained in client certificates.You can use one or a combination of these methods.

- Using SSL client authentication parameters as subdirectives, you can specify that the client certificate is valid or you can specify all or part of the Distinguished Name (DN) of a client or of the certification authority (CA) who issued the client's certificate.

  When you use SSL client authentication parameters, the server first checks to see if the client certificate is valid, as is. It compares any DN information in a protection setup and then compares any DN information in an ACL file with the DN information in the client's certificate. If the DN information matches, the server serves the document.

  You can use the Protection Setup Configuration and Administration form to specify the SSL client authentication parameters or you can specify the following on the Protection or Protect directive:

  – The validity of the client certificate.

    - SSL_ClientAuth: Use the SSL_ClientAuth subdirective with a value of *client* to require that all connections requesting documents under this protection setup must present an SSL certificate even if none of the other twelve SSL-specific subdirectives are present.

      If you are doing LDAP authentication, you can use the SSL_ClientAuth subdirective with a value of *client*. Doing so specifies that the certificate will suffice for authentication, without requiring a user ID and password.

      Syntax:

      ```
      SSL_ClientAuth client
      ```

  – All or any of the following subdirectives that make up a client Distinguished Name in the client certificate:

- CommonName - the common name of the client
- Country - the country in which the client resides
- Locality - the locality in which the client resides
- StateOrProvince - the state or province in which the client resides
- Organization - the organization of the client
- OrgUnit - the organizational unit of the client

– All or any of the following subdirectives that make up the CA Distinguished Name in the client certificate:
  - IssuerCommonName - the common name of the CA
  - IssuerCountry - the country in which the CA resides
  - IssuerLocality - the locality in which the CA resides
  - IssuerStateOrProvince - the state or province in which the CA resides
  - IssuerOrganization - the organization of the CA
  - IssuerOrgUnit - the organizational unit of the CA

Example

In the following example of an inline Protect directive, SSL client authentication must be set up, and the client must be making an **https** request and have a certificate with a common name of Dr Sheila A. Jones and an organization of RTP Quick Care Center to access the document.

```
Protect  /verysecret/*   {
CommonName="Dr Sheila A. Jones"
Organization="RTP Quick Care Center"
Mask  Anybody@(*)
}
```

- Code Distinguished Name fields in ACL files. For more information, see “Access Control List files” on page 174.
- Write a Go Webserver Application Programming Interface (GWAPI) program , usually an authorization or authentication exit, to extract Distinguished Name fields from the client certificate. Check these fields according to your own standards. Use this technique if you want to check a field in a way that you cannot with one of the client authentication subdirectives. For example, you may want to check part of the Organization field. Your GWAPI program extracts the HTTPS_CLIENT_CERT_ORGANIZATION environment variable for the Organization field, or the HTTPS_CLIENT_CERT environment variable for the certificate body. For more information on environment variables that extract Distinguished Name fields, see Appendix D, “Environment variables” on page 583.
- Use the UserCertfilter subdirective to specify which Distinguished Name fields in the client certificate the Web server searches on an LDAP server. Only one entry on the LDAP server can match the search. If more than one entry matches, the search fails.

  Example:

```
LDAPInfo SecureWay31c {
    Host abc.xyz.com
    ServerAuthType basic
    ServerDN "cn=ldapadm,ou=ihs test,o=ibm,c=us"
    ServerPasswordStashFile /usr/lpp/internet/server_root/ldap.sth
    ClientAuthType cert
    UserSearchBase "o=Deltaforce, c=US"
    UserCertfilter "(&(objectclass=person)(cn=%v1))"
    GroupSearchBase "o=Deltaforce, c=US"
}
Protection p17 {
    ServerId   "p17 LDAPc"
    PasswdFile "%%LDAP SecureWay31c%%"
```

```
    GroupFile  "%%LDAP SecureWay31c%%"
    Mask All
    SSL_ClientAuth client
}
```

For more information, see "Using LDAP to protect files" on page 298 and "LDAPInfo Subdirectives" on page 486.

- Cause System Authorization Facility (SAF) to verify the client certificate against a RACF user ID:
  - Map the client certificate to a RACF user ID by adding the client certificate to the RACF database and associating it with a RACF user ID.
  - Code `%%CERTIF%%` on the UserID directive, or the UserID subdirective

  For more information, see the client certificate examples in "Examples: Setting up secure connections" on page 69.

Notes:

1. Except for the option where you map the client certificate to a RACF user ID, you can choose which value you code on the UserID directive or UserID subdirective. Value options include `%%CLIENT%%`, `%%SERVER%%`, `%%CERTIF%%`, or a surrogate ID. The value that you choose indicates the user ID the Web server runs under when it handles a request. See "Userid - Specify the Default Access Control user ID" on page 457 for more information on the UserID directive and "UserID - Specify the Access Control user ID that the server should use" on page 448 for more information on the UserID subdirective.
2. The value you specify for the Mask subdirective varies, depending on whether you want the Web server to request a user ID and password. Generally, `Mask All` forces a prompt for a user ID and password. `Mask Anybody` does not. However, there is an interaction between the Mask subdirective, the PasswdFile subdirective, the AuthType subdirective, and the SSL_Client subdirective which can cause an unexpected result. Test each protection setup with a variety of inputs to make sure of the results. For example, test with a client certificate that contains the desired distinguished name information, another client certficate that does not contain the desired distinguished name information, a valid user ID and a valid password, an invalid user ID, a valid user ID and an invalid password, no user ID and no password, attempted access using the non-SSL port, and so on.

Hints and tips for coding SSL client authentication parameters for protection setups

- Specify any part of or all of a client or CA Distinguished Name.
- Enclose DN information that contains blanks or special characters in double quotes (as shown in the example).
- Make sure the DN information matches the DN information in the client certificate. This information is case sensitive and must have the same punctuation.
- Do not use wildcard characters for any of the parameters.

Step 5. Limit access to individual files

Perform this step only if you want to limit access to specific files on directories already protected by the protection setup.

To limit access to specific files on a protected directory, you create an Access Control List (ACL) file and place it on the directory. The ACL file must be named .www_acl.

Normally, the mask subdirectives in the protection setup define the first level of access control and then the ACL file further limits access. However, if you want all control to come from the ACL file, use the ACLOverride subdirective with a value of `On` in the protection setup. This causes the mask subdirectives in the protection setup to be ignored. All access control is passed to the ACL file.

See "Access Control List files" on page 174 for more information about ACL files.

Rules for specifying user names, group names, and address templates

Following are explanations and examples of the different ways you can specify user names, group names, and address templates on mask subdirectives. The same rules also apply for specifying user names, group names, and address templates in server group files, and ACL files.

- **User name without an address template:** You can specify a user name without an address template. The user name must be accessed through the `PasswdFile` subdirective as either a SAF user ID or a user name in a password file. If the requester returns the user name with the correct password, the server completes the request.

  ```
  GetMask  swandude
  ```

- **Address template without a user name:** You can specify an address template without a user name. If the requester address matches the address template, the server completes the request without prompting the requester for a user name and password.

  The address template can be based on either IP address or host name. Use the asterisk character (*) as a wildcard in any part or the template. To indicate you want to use address template protection only, precede the template with one of the following: `@`, `Anybody@`, `Anyone@`, or `Anonymous@`.

  ```
  GetMask    Anybody@123.45.2.*
  PostMask   @96.*.*.*
  DeleteMask Anonymous@walden.pond.*.*
  ```

  Note: In order to compare the requester host names against address templates, you must set the DNS-Lookup directive to `On`. If the DNS-Lookup directive is set to `Off` (the default), your server can compare only the IP address of the requester to the address templates. See "DNS-Lookup - Specify whether you want to look up host names of clients" on page 450.

- **User name with an address template:** You can specify a user name with an address template. The user name must be accessed through the `PasswdFile` subdirective as either a SAF user ID or a user name in a password file. Separate the user name from the address template with the at sign character (@). If the requester returns the user name with the correct password and the requester address matches the address template, the server completes the request.

  ```
  GetMask   webfoot@96.96.*.*
  PostMask  billface@*.ibm.com
  ```

- **User names of All or Users:** You can use the value `All` or `Users` with or without an address template to represent all user names defined in either a SAF security database or the password file. If the requester returns any user name and password defined in the protection setup password file, and the requester address matches any address templates, the server completes the request. The server completes the request if the following requirements are met:
  - The requester address matches any address templates.
  - The requester returns any valid user ID and password. The user ID and password would be checked based on information defined on the `PasswdFile`

subdirective of the `Protection` directive. If `PasswdFile` is defined with `%%SAF%%`, then the user ID and password would be checked against a SAF security database. If `PasswdFile` is defined with an actual password file, the user ID and password would be checked against those user IDs and passwords listed in the password file.

```
GetMask    All
PostMask   All@(96.*.*.*,*.ibm.com,123.45.2.*)
```

- **Group name:** You can specify a group name that is defined in the server group file specified on the GroupFile subdirective (see "Group files").

  A group name can include user names, other group names, and address templates in any of the same formats allowed on masking subdirectives. In order for any user names in the group to be valid, they must be defined as either a SAF user ID or a user name in a password file If the requester returns a valid user name and password and any address templates are matched, the server completes the request.

  ```
  GetMask   ducks
  PostMask  geese
  ```

- **Anybody, Anyone, or Anonymous:** You can use the values `Anybody`, `Anyone`, or `Anonymous` without an address template or with an address template of `(*)` to indicate you do not want to use any protection for requests covered by the subdirective. The server completes requests without prompting for a user name and password and without checking the address of the requester.

  ```
  GetMask  Anybody
  PutMask  Anyone@(*)
  ```

- **Multiple items on a subdirective:** You can specify multiple items on each subdirective. Separate each item with a comma. The comma is treated as a logical or.

  ```
  GetMask   swandude@96.96.*.*,geese,ducks,Anyone@walden.pond.*.*
  ```

- **User or group names spanning multiple lines:** You can continue a list of user and group names onto a new line by ending the previous line with a comma.

  ```
  GetMask   swandude@96.96.*.*,webfoot@water.fowl.com,
            geese,billface
  ```

- **Parentheses for grouping user names, group names, and address templates:**You can use parentheses to keep user names and group names together or address templates together.

  ```
  GetMask   (gooduser,webfoot)@96.96.*.*,
            billface@(water.fowl.com,123.45.2.*),
            (goosegg,bagel,elnada)@(98.*,146.*)
  ```

Group files

Use group files to classify users into groups. A group file can consist of one or many group records. Each record defines a single group. A group can consist of other groups previously defined in the group file, or individual users.

You can create as many server group files as you need. Create each in a separate text file. Within the group file, each line contains a group definition using the following format:

```
groupname : user1[,user2[,user3...]]
```

groupname
: Any name you want to use to identify the group you are defining. This name can be used on:

- Mask subdirectives within protection setups that point to the server group file. (The mask subdirectives are DeleteMask, GetMask, Mask, PostMask, and PutMask.)
- Access rules within ACL files on directories that are protected with a protection setup that points to the server group file.
- Subsequent group definitions within the same server group file.

user1[,*user2*[,*user3...*]]
This can actually be any combination of user names, group names, and address templates. Separate each item with a comma.

For non-SAF user names to be valid, they must be defined in the password file that the protection setup points to. You create and maintain password files using the htadm command.

Note: The htadm command cannot be used to manage SAF user IDs and passwords.

Group names must be defined on previous group definition statements in the same group file.

Note: Do **not** put blanks in any of the names specified on the user fields as unpredictable results may occur.

Example 1

```
ducks  : (webfoot,billface)@96.96.3.1,swandude
geese  : goosegg,bagel@(walden.pond.*.*,123.*.*.*)
flock  : ducks,geese
webbed : All@water.fowl.*
fish   : salmon,trout,bass
```

In this example, notice that the groups named `ducks` and `geese` are defined prior to being declared as members of the group called `flock`. For an explanation of how to specify user names, group names, and address templates, see "Rules for specifying user names, group names, and address templates" on page 171.

Use of group files in protection setups

Protection setups can point to a Web server group file. The protection setup can then use the groups defined in the group file on mask subdirectives. If a protected directory contains an Access Control List (ACL) file, the rules in the ACL file can also use the groups defined in the group file.

Note: Only groups defined in a group file can be used with the `Protection` directive. SAF groups cannot be used.

Example 2

This example demonstrates the use of the GroupFile subdirective on a Protection directive. The user ID of **user123** is allowed to gain access to a URL protected by /secret/*. In order to gain access, the user must have a valid SAF user ID and password and must be in the group named `group1`.

1. Create the group file`/u/mydir/my.grpfile` using a text editor:

   ```
   group1: user123
   ```

   where

   - **group1** is the group name.

- **user123** is a group member. In this example **user123** is a valid SAF user ID.
- **/u/mydir/my.grpfile** is an HFS file.

2. Define the protection setup in the httpd.conf file:

```
Protection MYprot {
ServerID   MYSTUFF
UserID     %%CLIENT%%
PasswdFile %%SAF%%
GroupFile  /u/mydir/my.grpfile
Mask       group1
}

Protect    /secret/*   MYprot
```

The subdirective **PasswdFile** will cause **user123** to be prompted for a valid SAF user ID and password. The user ID and password will be checked in the security database. **user123** will pass the check as long as a valid password is provided. Any valid SAF user ID on the system can pass the security check. However, a user ID cannot gain access to the files under **/secret/*** unless it is in the group specified on the **Mask** subdirective. That group is called **group1** and is in the file specified on the **GroupFile** subdirective. In the previous step we defined **group1** in the file **/u/mydir/my.grpfile**. **user123** is the only user ID in **group1**, and therefore, the only user ID that can access files in **/secret/***. All other users will be denied access.

Note: If there is a problem with the `GroupFile`, the most likely indication will be an error of `401 NO ACCESS, IMW0216E NOT AUTHORIZED, AUTHENTICATION FAILED`. This message can appear in the client's browser, in the Web server error log, or both.

Note: `GroupFile %%LDAP%%` is **not** covered by this explanation.

Access Control List files

Use Access Control List (ACL) files to limit access to specific files on a protected directory. Note that using ACL files can impact performance.

Each protected directory can have only one ACL file. The ACL file must be named .www_acl and be present on the protected directory. Normally, the mask subdirectives in the protection setup define the first level of access control and then the ACL file further limits access to individual files. However, if you want all control to come from the ACL file, use the ACLOverride subdirective with a value of `On` in the protection setup. This causes the mask subdirectives in the protection setup to be ignored when a protected directory contains an ACL file.

Within the ACL file, each line contains a rule limiting access based on file name, HTTP method, and authorized users, groups, or addresses. The server processes the rules in the ACL file from top to bottom. The server compares the three elements of each rule to the request until a match is found or until the end of the file is reached.

The format for rules in ACL files is:

file : *method* : [*SSLauth*,] *user*

file A file name or a template for the files you want to protect with this rule. Each file template can contain one asterisk (*) wildcard character.

The server denies any requests for files that are not listed or covered by a template.

method **or** *method1*[**,***method2*[**,***method3...*]]
An HTTP method or list of methods, separated by commas. The methods define what kind of requests the authorized users are allowed to make for the protected files. Any method you specify must also be enabled. See "Methods - Set method acceptance" on page 518 for an explanation of the HTTP methods supported by the server and how to enable them.

SSLauth
If you have implemented SSL client authentication, you can code ACL files that specify any or all parts of the Distinguished Name of a client or the CA who issued the client's certificate. You can also include more than one client or CA parameter in a given ACL. The server requests the client's certificate. The server compares the DN information in the certificate first to DN information in protection setups and then to DN information in ACL files. If the DN information matches, the server returns the document.

The ACL can specify any or all of the following parts of the

- Client Distinguished Name in the client certificate:
  - !CommonName - the common name of the client
  - !Country - the country in which the client resides
  - !Locality - the locality in which the client resides
  - !StateOrProvince - the state or province in which the client resides
  - !Organization - the organization of the client
  - !OrgUnit - the organizational unit of the client
- CA Distinguished Name in the client certificate:
  - !IssuerCommonName - the common name of the CA
  - !IssuerCountry - the country in which the CA resides
  - !IssuerLocality - the locality in which the CA resides
  - !IssuerStateOrProvince - the state or province in which the CA resides
  - !IssuerOrganization - the organization of the CA
  - !IssuerOrgUnit - the organizational unit of the CA

user **or** *user1*[**,***user2*[**,***user3...*]]
This can actually be any combination of user names, group names, and IP address templates. Separate each item with a comma.

For user names to be valid, they must be defined in the password file that the protection setup points to. Group names must be defined in the server group file the protection setup points to.

The items you specify have to follow the same rules as you use to specify user names, group names, and address templates on mask subdirectives. See "Rules for specifying user names, group names, and address templates" on page 171.

Examples

In the following example, any valid user name and password can be used to GET any file on the directory. User names defined for the group geese can be used to POST to any HTML files. From IP address 123.34.14.2, the user name webfoot can be used to DELETE the welcome.html file.

```
*             : GET             :  All
*.html        : GET,POST        :  geese
golden.*      : GET,POST        :  geese,@bean.stalk.*
welcome.html  : GET,POST,DELETE :  webfoot@123.34.14.2
```

In the following example, SSL client authentication must be set up, and the client must be making an **https** request from an IP address beginning with 9.67 and have a client certificate with a common name of Dr Shelia A. Jones to access the patientA.html document.

```
patientA.html : GET : !CommonName="Dr Shelia A. Jones"@9.67.*.*
```

In the following example, SSL client authentication must be set up, and the client must be making an **https** request from an IP address beginning with 9.67 and have a client certificate with a common name of Dr Shelia A. Jones or Dr Harry S. Smith to access the patientA.html document.

```
patientA.html : GET :
  !CommonName=("Dr Shelia A. Jones","Dr Harry S. Smith")@9.67.*.*
```

Note: In this example the entry appears on two lines for printing purposes. Each entry appears on one line in your file.

Secure Sockets Layer client authentication hints and tips

The following hints and tips apply if you code SSL client authentication parameters in an ACL file:

- Specify any part of or all of a client or CA Distinguished Name.
- Specify the DN information for multiple clients or CAs on a given parameter. The DN information should be separated by a comma and enclosed in parentheses (as shown in the above example).
- Enclose DN information that contains blanks or special characters in double quotes (as shown in the above example).
- Make sure the DN information matches the DN information in the client certificate. This information is case sensitive and must have the same punctuation.
- Check that all of the ACL rule is on one line.
- Do not use wildcard characters for any of the parameters.
- If you enable SSL client authentication, but you have an empty ACL file, noone has access to the protected Web pages. Resolve the problem in one of two ways:
  - Put at least one part of the Distinguished Name in the ACL file.
  - Remove the ACL file and put at least one part of the Distinguished Name in the protection setup.

Default protection scheme when coding %%CLIENT%% on UserID directive

When you define the UserID directive with `%%CLIENT%%`, the Web server implements the following Protect directive, by default. The Protect directive contains an in-line protection setup.

```
Protect  *    {
   ServerID   System_Logon
   UserId     %%CLIENT%%
   AuthType   Basic
   PasswdFile %%SAF%%
   Mask       All
}
```

Most browsers present the name specified on the ServerID so users know which protection scheme they must provide a given user ID and password for. With the default implementation, browsers present the name System_Logon to users. You can present a name other than System_Logon. Code a Protect directive as shown for the default Protect directive, but change the name on the ServerID subdirective to the desired name.

Note: If you omit the ServerID subdirective, the Web server creates a default name of `UNKNOWN`.

The Web server uses the last Protect directive on which a URI matches the request template. Since the request template on this Protect directive is an asterisk, any URI can match this request template. Put this Protect directive before all other Protect directives so that URIs match the more specific request templates last.

Examples: Protecting server resources

Example 1: Setting up protection with user names and passwords

This example shows how to protect server resources by prompting users for a valid user ID and password.

Step 1. Accept the Web server default user ID %%CLIENT%%

The sample configuration file provided with the Web server protects the default home page and samples with %%CLIENT%% so that only users known to the system can use the server until you explicitly grant access to other users.

For more information on %%CLIENT%% and other access control user ID options, see “Userid - Specify the Default Access Control user ID” on page 457.

Step 2. Create a URL request template for the resource that needs to be protected and specify protection settings for that request

When a client sends a request for a resource to the Web server, the server uses the content of the request to activate protection for the resource. A request is the part of a URL that follows the server’s host name. For example, the host name of the IBM Web site is www.ibm.com. If a client enters URL http://www.ibm.com/Products, the request is /Products.

In this example, Company A wants to ensure that requests for the budgets.html file located in the financial directory are protected. To restrict access to the budgets file, Company A needs to create a URL request template and specify protection settings for the request, /financial/budgets.html, using these steps:

1. From the default Front Page of the Web server, click Configuration and Administration Forms.
2. Click Access Control, then click Document Protection.
3. Accept the default position in the list, Insert before.
4. For URL request template, enter /financial/*. This restricts access to all files in the financial directory, including budgets.html.
5. For Authentication options, click Password or user/group authentication.

   To use the SSL client authentication option, you must set up secure communications to and from your server. For examples, see “Examples: Setting up secure connections” on page 69.

6. Click Named protection setup and enter BUDGETS for the protection setup name.

   If you define the protection setup in-line, you can only use the settings for this one URL request template. A named protection setup is recommended because it can be used for other URL request templates.
7. Leave the Server IP Address or host name field blank.

   This field is used if your server has multiple IP addresses.
8. Click Submit.
9. For performance reasons, we do not select the option to Allow ACL files to override protection settings. For more information on ACL files, see "Access Control List files" on page 174.
10. Enter your Password Authentication settings:
    - For Protection realm, enter restricted.

      Protection realm is a name that you select to identify this protection setup to requesters who will be entering a user ID and password. The name does not need to be a real machine name. When the server sends a requester a prompt for user name and password, it also includes the name you specify for protection realm. Protection setups that use the same password should use the same name for protection realm.
    - Enter %%SAF%% for Password file.

      When %%SAF%% is specified, password verification is performed using your SAF-based security product, for example, RACF.
    - Leave the Group file field blank.

      This field is used only if you specify group names in the Permissions section.

      Group files cannot contain RACF user IDs. For more information on group files, see "Group files" on page 172.
    - For Permissions, give Read permission only to those users who are authorized to access budget information. Enter each user name, separated by a comma, in the Users with Read permission (GET and POST) field.
11. Click Submit. The new URL request template appears in the list.
12. Click Configuration Page to configure additional options.

Step 3. Set up mapping rules to route the request to the correct resource

Before your Web server can route a request to a specific resource, you need to configure mapping rules for the server to use. To route requests for the budgets.html file to the correct directory (financial):

1. Click Request Processing, then click Request Routing.
2. Accept the default position in the list, Insert before.
3. For URL request template, enter /budgets.html/.
4. Click Submit. You will see a message that the requested configuration changes have been completed successfully.

Step 4. Restart the server to use your new configuration settings

Example 2: Setting up a protection setup without SSL client authentication

Following are examples of the files you can use for protecting your server resources. In the files where they are allowed, comments begin with the pound sign (#) in the first column.

Following is an example of the protection related portion of a server's configuration file.

```
Protection    POND-PROT {
  ServerID    Feathered
  Authtype    Basic
  PasswdFile  /etc/flying.pwd
  GroupFile   /etc/nesters.group
  GetMask     All@*.swimmer.org
  PutMask     quacks,billface
  DeleteMask  bigbird@pond.hq.swimmer.org
}
#
# The above Protection directive defines a protection setup named
# POND-PROT.  The Protection directive must be placed before any
# Protect or DefProt directives that point to it. The Mask
# subdirectives restrict access as follows:
# o Any requester from a host name ending with .swimmer.org can
#   GET files.  They must be able to enter any user name and
#   password defined in the /etc/flying.pwd password file.
# o To PUT files, the requester must match the restrictions
#   defined for the group named quacks in the
#   /etc/nesters.group server group file.
#   Or, the requester must enter the user name billface and the
#   password defined for it in the C:\TCPIP\etc\flying.pwd
#   password file.
# o To DELETE files, the requester must be from the host named
#   pond.hq.swimmer.org.  The requester must enter the user name
#   bigbird and the password defined for it in the
#   /etc/flying.pwd password file.
#
Protect   /wetland/creatures/*   POND-PROT
#
# Any request beginning with /wetland/creatures/
# activates protection as defined in the protection setup
# labeled POND-PROT
#
Protect   /vegetation/*   /usr/lpp/internet/server_root/webster.setup
#
# Any request beginning with /vegetation/ activates protection
# defined in the separate protection setup file named
# /usr/lpp/internet/server_root/webster.setup
#
Protect   /flocks/*   {
  DeleteMask   @9.67.84.5
  MASK         Anybody@9.67.*.*
}
#
# The above Protect directive contains an in-line protection setup.
# Any request beginning with /flocks/ activates protection
# defined as part of the Protect directive.  The mask subdirectives
# restrict access as follows:
# o To delete files, the requester must be at the host with
#   IP address 9.67.84.5
# o To use any other enabled HTTP method, the requester must
#   be at a host with an IP address beginning with 9.67.
# (@ by itself and Anybody@ both mean the same thing.)
# Since only address protection is being used, there is no need for
# the Authtype, ServerID, PasswdFile, and GroupFile subdirectives.
#
Pass  /wetland/*  /usr/lpp/internet/server_root/pub/freshwater/*
```

```
#
# For requests beginning with /wetland/, the server goes to
# the /usr/lpp/internet/server_root/pub/freshwater/ directory
# to find the file.
#
Pass  /*   /usr/lpp/internet/server_root/pub/*
#
# For any requests not matching other Pass directives, the server
# goes to the /usr/lpp/internet/server_root/pub/ directory (the
# document root directory).  For example, if the server received
# the request /vegetation/cattail.html, it would look for the
# cattail.html file on /usr/lpp/internet/server_root/pub/vegetation.
#
```

Following is an example of what the /usr/lpp/internet/server_root/webster.setup file might look like:

```
ServerID    Webby
Authtype    Basic
PasswdFile  /etc/flying.pwd
GroupFile   /etc/nesters.group
GetMask     All@(123.45.*,water.*)
PutMask     goosegg,webfooter,nest
DeleteMask  (swandude,billface)@(water.fowl.com,123.45.2.*)
ACLOverride On
#
# The mask subdirectives restrict access based on a combination of
# user names and passwords and address templates.  However, since
# ACLOverride is on, these subdirectives are ignored and the ACL
# file is used.  For user name and password protection, the ACL file
# must use the /etc/flying.pwd password file and the
# /etc/nesters.group server group file.
#
```

Following is an example of what the /etc/flying.pwd file might look like:

```
bigbird:vfzDlIeUzFzCt:The big guy
duckman:PVz1Y6YFI8IY3:The duck
swandude:FexekemFhY7q8:Swanee
billface:ZereWePF0YJqK:Orange nosed one
goosegg:dh*ySyZJPOqrw:Shut out
```

Note: When you look at the contents of the password file, you cannot see the passwords because they are encrypted. To manage the password file you must use the htadm command. Do not edit password files with a text editor.

Following is an example of what the etc/nesters.group file might look like:

```
ducks  : (webfoot,billface)@96.96.3.1,swandude
geese  : goosegg,bagel@(walden.pond.*.*,123.*.*.*)
quacks : ducks,geese,Anybody@43.234.*
nest   : All@(water.fowl.*,nesting.*.*)
#
# The quacks group demonstrates how you can use previously
# defined groups in a subsequent group definition.  The quacks
# group includes both the ducks and geese groups.  Additionally,
# the quacks group includes address template protection for any
# requests from hosts with an IP address beginning with 43.234.
# Requests from matching hosts would not be prompted for a user
# name and password.
```

Following is an example of what the .www_acl file on /usr/lpp/internet/server_root/pub/vegetation might look like.

```
*             : GET             :  All
*.html        : GET,POST        :  nest
billed.html   : GET,POST        :  duckman,geese
welcome.html  : GET,POST,DELETE :  bigbird
#
# All restrictions come from the ACL file because the protection
# setup specified ACLOverride On.  The user names and groups used
# in the ACL file must be defined in the password and server group
# file identified in the protection setup.
```

Example 3: Setting up a protection setup with SSL client authentication

You can restrict who can access documents by using password files and/or user or group authentication in protection setups as shown in the examples in "Example 2: Setting up a protection setup without SSL client authentication" on page 178. You can further restrict who can access documents by coding SSL client authentication parameters in protection setups, ACL files, or both.

The following examples assume that SSL client authentication has been set up by specifying SSL client authentication on the Security Configuration form or by coding the SSLClientAuth directive in the configuration file.

```
Protection SSL_CLIENT_AUTH {
   CommonName "Dr Sheila A. Jones"
   Country US
   Organization "RTP Quick Care"
   Mask  Anybody@(*)
}
```

For a server to serve the document, the server authenticates the client by requesting and ensuring the client has a valid certificate. In this case, for the certificate to be valid, the Distinguished Name of the client must be made up of a Common Name of Dr Sheila A. Jones, an organization of RTP Quick Care Center, and a country of US.

If you want to allow all persons with an organization unit of Pediatrics to access the documents protected by th SSL_CLIENT_AUTH protection setup, you could specify:

```
Protection SSL_CLIENT_AUTH {
   OrgUnit Pediatrics
}
```

The Protect directive that maps the request to the protection setup follows the protection setup:

```
Protect /secure_docs/* SSL_CLIENT_AUTH
```

Any request beginning with secure_docs activates protection as defined in the protection setup SSL_CLIENT_AUTH.

Note: If you code SSL client authentication parameters in ACL files and not in protection setups, you must also have a Protect statement like the one in the above example.

Changing expired passwords

When you require a user to provide a user ID and password, the password that the user types in can be valid, invalid, or expired. The HTTP protocol uses the same 401 response code to indicate both an invalid password and an expired password. The user sees the message indicating that his password has expired only when he presses *cancel* on the pop-up browser window.

Here are the steps for a user to change an expired password:

1. At the password pop-up window, type
   - Your user ID in the user ID field.
   - *<old_password>/<new_password>/<new_password>* in the password field. Note that the user **must** separate the old and new passwords by a forward slash.
2. Click OK.
3. The password pop-up window will appear again. Type in your user ID and the new password.

You can alternatively implement a sample gwapi function called pwapi.c. This sample identifies when a user's password has expired. It then provides a form for changing the password. You can find the sample in the WebSphere Troubleshooter for z/OS and OS/390.

To link to the *Troubleshooter*, go to URL:

`http://www.ibm.com/software/webservers/httpservers/troubleshooter.html`

Part 5. Advanced Configuration

Chapter 10. Customizing cache management with the Fast Response Cache Accelerator

Overview 185
Cache Accelerator eligibility 185
Planning considerations 186
Access to z/OS Workload Management (WLM) 186
Unique subsystem name for WLM 186
TCP/IP stack name 186
Enabling and configuring the Cache Accelerator 186
Monitoring and managing the Cache Accelerator 187
RMF reports 187
DISPLAY TCPIP command 187

Overview

The Fast Response Cache Accelerator can improve the performance of the HTTP Server when serving text and image files.

Because the Cache Accelerator cache is automatically loaded during server operation, you are not required to list the files to be cached in your server configuration file. In addition, the server will automatically recache changed pages and remove outdated pages from the cache.

The Cache Accelerator provides support for caching on multiple Web servers and on servers with multiple IP addresses. Currently, support is not available for running the Cache Accelerator on a proxy server.

Cache Accelerator eligibility

Requests for the following content are **not** eligible for inclusion in the Cache Accelerator:

- Dynamically generated content.
- Files containing meta-information.
- Pages generated using Server Side Includes.
- Objects served from proxy requests.
- Objects served over an SSL connection.
- URLs specified on the`FRCANoCaching` directive.
- URLs that are not specified on the `FRCACacheOnly` directive, if the `FRCACacheOnly` directive is defined.
- Files protected using the Protect directive unless the value on one of the mask subdirectives is `Users`, `All`, `Anybody`, `Anyone`, `Anonymous`, or `*`.

  Note: If a mask subdirective such as `PostMask` returns dynamic content, the content will not be eligible for caching even if `Users`, `All`, `Anybody`, `Anyone`, `Anonymous`, or `*` is coded on the mask subdirective.
- Files requested when the `UseACLs` directive is set to `always`. When the directive is set to `protectonly`, files that are not covered by a protection statement are eligible for caching.
- Files requested when the Web server is configured for virtual hosts, and the host name header is omitted from the client request.

Once a file is loaded into the Cache Accelerator, it is accessible without requiring authentication. You must explicitly configure the Web server to prevent files that require authentication from being loaded into the Cache Accelerator.

Example

- The UserID directive for web site ABC is set to **%%CLIENT%%**.
- The Cache Accelerator is enabled.
- The home page of web site ABC is eligible for caching.
- User JSMITH requests the home page of web site ABC.
- JSMITH is prompted for his user ID and password. He provides it.
- JSMITH receives the home page of web site ABC.
- The Cache Accelerator places a copy of the home page into its cache.
- User JDOE requests the home page of web site ABC.
- JDOE receives the home page of web site ABC even though he was not prompted for his user ID and password. (The home page was served from the Cache Accelerator's cache.)

To insure that JDOE is prompted for a user ID and password, make sure that the home page is ineligible for caching. You could, for example, define the home page of web site ABC on the `FRCANoCaching` directive.

Planning considerations

Access to z/OS Workload Management (WLM)

Before using the Cache Accelerator, ensure that the HTTP Server has access to WLM as outlined in "z/OS Workload Management" on page 27.

Unique subsystem name for WLM

If you are not using the -SN parameter when starting the server, you must specify a unique subsystem name for the Cache Accelerator when you configure dynamic caching. For more information, see the description of the FRCAWLMParms directive in "Tuning - Define performance and scalability settings" on page 566.

TCP/IP stack name

If you are using common INET and multiple TCP/IP stacks, identify the name of the TCP/IP stack which supports the Cache Accelerator. This name must match the name on the SubFileSysType statement in the z/OS UNIX BPXPRMxx parmlib member. For more information, see the description of the FRCAStackName directive in "Tuning - Define performance and scalability settings" on page 566.

Enabling and configuring the Cache Accelerator

By default, the Cache Accelerator is not enabled.

You can enable and configure the Cache Accelerator by editing the Web server configuration file directly or by using the Configuration and Administration Forms:

- To edit the configuration file:
  1. Open the httpd.conf configuration file.
  2. Set the EnableFRCA directive on.
  3. To configure the Cache Accelerator, use the FRCA directives. For information on FRCA directives, see "EnableFRCA — Turn dynamic caching on or off" on page 567.
- To use the Configuration and Administration Forms:
  1. From the Front Page of your server, click **CONFIGURATION AND ADMINISTRATION FORMS**.

2. Click **Fast Response Cache Accelerator**.
3. Click **Enable Fast Response Cache Accelerator** to turn dynamic caching on.
4. Configure the Cache Accelerator using the appropriate options. Refer to the online help for more information on each option.
5. Restart the server to use the new settings.

Monitoring and managing the Cache Accelerator

Two resources for monitoring and managing the Cache Accelerator are Resource Measurement Facility (RMF) reports and output from the DISPLAY TCPIP command.

RMF reports

The WLM enclave created for the Cache Accelerator is a scheduled SRB. Therefore, to manage the Cache Accelerator workload, you need one service class period that has an execution velocity goal. The RMF report will show no transactions, and CPU information will be included under TCB CPU consumption.

For more information on RMF, see the *RMF User's Guide*.

DISPLAY TCPIP command

To display information on the Cache Accelerator enter:

D TCPIP,*FRCA_stack_name***,NET,CACH**

For *FRCA_stack_name*, specify the name of the z/OS UNIX physical file system that supports the TCP/IP stack used by the Cache Accelerator, for example, TCPV34. This is the name that appears on the FRCAStackName directive in the server configuration file.

Sample output:

```
D TCPIP,TCPV34,NET,CACH
EZZ2500I NETSTAT CS V2R7 TCPV34 324
CLIENT: WEBSRV3 LISTENING SOCKET: 0.0.0.0..8137
MAXCACHESIZE:       0000002500 CURRCACHESIZE:      0000000120
MAXNUMOBJECTS:      0000000200 CURRNUMOBJECTS:     0000000045
NUMCONNS:           0000002116 CONNSPROCESSED:     0000001576
CONNSDEFERRED:      0000000540 CONNSTIMEDOUT:      0000000000
REQUESTSPROCESSED:  0000002116 INCOMPLETEREQUESTS: 0000000000
NUMCACHEHITS:       0000001576 NUMCACHEMISSES:     0000000540
NUMUNPRODCACHEHITS: 0000001576
1 OF 1 RECORDS DISPLAYED
```

For more information on DISPLAY TCPIP output fields, see the *TCP/IP OE User's Guide*.

Chapter 11. Customizing logs and reports

Tailoring the logs your server keeps 189
Overview of log types 189
Server Access log 189
FRCA Access log 190
Proxy server access logs 190
Agent and Referer logs 190
Server Error log 190
CGI Error log 191
Overview of log setup options. 191
Specifying global settings for all logs . . . 191
Specifying the path and file name for the Access, Agent, and Referer logs 191
Specifying the path and file name for the FRCA Access log 192
Specifying the path and file name for the proxy server's Cache Access log 192
Choosing log maintenance options for the Access, Agent, and Referer logs 192
Setting filters for the Access, Agent, and Referer logs 193
Specifying options for the error logs. . . . 194
Sample scenario for configuring log files . . . 196
Forms 196
Directives 196
Tailoring the reports your server creates 196
Reporting options and considerations 196
Configuring the Web server to use a third-party reporting program 197
Proxy server considerations 197
Using the HTLOGREP or Web usage mining reporting programs 197
Creating a report template 199
Viewing reports 200
Sample scenarios for configuring reports . . . 200
Sample report: Top 100 page hits 200
Sample report: PUT requests to beta subdirectory 201
Sample report: Accesses, excluding beta subdirectory and alpha7 requests 202
Sample report: Accesses for department server and for the beta subdirectory except Alpha7*.* files 202
Using the Web usage mining statistics reports . . 203
Running Web usage mining from the command line. 205
Web usage mining and multiple servers . . . 206
Logging information with System Management Facilities 206

This chapter explains how to tailor the server's access and error logs to meet your needs, and how to create customized reports from the information in the logs.

Tailoring the logs your server keeps

Overview of log types

The HTTP Server creates the following types of logs:

- Server Access log
- FRCA Access log, if you are using the Fast Response Cache Accelerator for dynamic caching
- Proxy and Cache Access logs, if your server is running as a proxy
- Agent log
- Referer log
- Server Error log
- CGI error log

Server Access log

The server logs activity in the ***access log*** files and stores them each night. At midnight each night, the server closes the current access log and creates a new access log file for the coming day. The access log contains entries for page request mode to the server.

For each access request your server receives, an entry is made in the access log showing:

- What was requested
- When it was requested
- Who requested it
- The method of the request

- The type of file that your server sent in response to the request
- The return code, which indicates whether the request was honored

FRCA Access log

The ***FRCA Access log*** is an optional log.

Set up the FRCA Access log if you want requests served by the Fast Response Cache Accelerator to be logged in a separate log file. If this log is used, no other log entries will be written for requests served from the Cache Accelerator cache. If this log is not used, requests that are served by the Cache Accelerator will be logged the same as other requests.

You may want to use this optional log if you are concerned about performance but still want access log information about requests served from the Cache Accelerator cache.

The server logs activity in the FRCA Access log file and stores it each night. At midnight each night, the server closes the current access log and creates a new access log file for the coming day.

For each access request your server receives, an entry is made in the access log showing:
- What was requested
- When it was requested
- Who requested it
- The method of the request
- The type of file that your server sent in response to the request
- The return code, which indicates whether the request was honored

Related Information:

- For information on the dynamic caching function, see Chapter 10, "Customizing cache management with the Fast Response Cache Accelerator" on page 185.

Proxy server access logs

If your server is running as a proxy, the server can create two different types of logs:
- A ***proxy access log***, which contains access requests for files that come from the proxy server
- A ***cache access log***, which contains access requests for files that come from the proxy server's cache

Agent and Referer logs

The ***agent log*** indicates which Web browser was used to access a Web page. The ***referer log*** identifies the Web page that referred (or linked to) the requested Web page. By default the server writes an entry to the agent and referer logs each time a client sends the server a request. For every entry made in the access log:

- The agent log has a corresponding entry that indicates the browser used to display the page or file requested by the client
- The referer log has a corresponding entry that indicates the referring page

Server Error log

The server creates an ***error log*** that includes errors encountered by your server's clients, such as timing out or not getting access.

CGI Error log

The server creates a ***CGI error log*** that logs standard error output (stderr) from CGI programs.

Overview of log setup options

To set up the logs to suit your particular needs, refer to the following sections:

- "Specifying global settings for all logs"
- "Specifying the path and file name for the Access, Agent, and Referer logs"
- "Specifying the path and file name for the FRCA Access log" on page 192
- "Specifying the path and file name for the proxy server's Cache Access log" on page 192.
- "Choosing log maintenance options for the Access, Agent, and Referer logs" on page 192
- "Setting filters for the Access, Agent, and Referer logs" on page 193
- "Specifying options for the error logs" on page 194

Note: You can change the default settings for the logs either by using the online Configuration and Administration forms or by manually editing the directives in the configuration file.

Specifying global settings for all logs

In most cases, you will want to accept the default global settings, which apply to all logs.

If you plan to use the reporting functions described under "Tailoring the reports your server creates" on page 196, you must accept the default file format, ***common.***

If you want to have log information sent to the HTTP Server window in addition to sending it to the log files, you must change the default.

To change the global settings, we recommend that you specify them on the **Global Log File Configuration** form.

Defaults: Common file format, which is used by most Web servers, and local time format are used. Log files are written nightly.

Directives:

- For time stamp, edit the LogTime directive. For more information, see "LogTime - Specify GMT or local time stamps in log files" on page 510.
- For file format, edit the LogFormat directive. For more information, see "LogFormat - Specify common or old log file format" on page 506.

You cannot change whether the logs are written nightly.

Specifying the path and file name for the Access, Agent, and Referer logs

Note: If you are using a separate access log for the Fast Response Cache Accelerator, see the instructions in "Specifying the path and file name for the FRCA Access log" on page 192.

From the **Access Log File Configuration** form, you can specify the path and name of the directory where you want to place the access, agent, and referer log files.

Directives:

- For the access log path, edit the AccessLog directive. For more information, see "AccessLog - Name the path for the access log file" on page 492.
- For the agent log path, edit the AgentLog directive. For more information, see "AgentLog - Name the path for the agent log file" on page 502.
- For the referer log path, edit the RefererLog directive. For more information, see "RefererLog - Name the path for the referer log file" on page 512.

Specifying the path and file name for the FRCA Access log

From the **Fast Response Cache Accelerator** form, you can specify the path and file name for the FRCA Access log file.

Directive: For the FRCA access log path and name, edit the FRCAAccessLog directive. For more information, see "FRCAAccessLog — Specify a log file path and name for dynamic caching requests" on page 569.

Specifying the path and file name for the proxy server's Cache Access log

If the server is running as a proxy, you can log requests to the cache separately from other requests.

From the **Access Log File Configuration** form, you can specify the path and file name where you want the server to put access requests that are satisfied from the proxy server's cache. As an alternative, you can specify this information manually by editing the directive listed below.

Directive: For the Cache Access log path, edit the CacheAccessLog directive. For more information, see "CacheAccessLog - Specify the path for the cache access log files" on page 502.

Choosing log maintenance options for the Access, Agent, and Referer logs

Note: If you have set up a FRCA Access log for the Fast Response Cache Accelerator, your log maintenance options will apply to that access log.

With the log maintenance options, you can specify how to handle the accumulation of daily logs for days past.

You can choose whether you want to keep old logs, remove logs after they reach a certain age and/or a collective size, or run your own program at midnight each night to handle old logs. Note that the "collective size" is the collective size of all access logs only (not combined with agent and referer logs), or of all agent logs only (not combined with access and referer logs), or of all referer logs only (not combined with access and agent logs).

To reduce the space the access, agent, and referer logs require, you can specify that the logs be automatically removed, based on the age of the log and/or the collective size of the logs.

If you are interested in running your own backup program to store the logs, you can specify a user exit. In this case, you specify the path to your program and the parameters to pass to your program. The server appends to this information the path to the logs.

We recommend you define these options on the **Access Log File Configuration** form, but you can edit the configuration file to include the appropriate directives. The settings you specify on the **Access Log File Configuration** form apply to agent and referer logs, as well.

Defaults: By default, all access, agent, and referer log files are kept at the path location you specify on the **Access Log File Configuration** form (or the AccessLog, AgentLog, and RefererLog directives).

Directives: The directives you specify for access logs apply to agent and referer logs, as well.

- To keep access log files, no directive is required. `AccessLogArchive none` is the default.
- To remove access log files based on age, edit these directives:
  - `AccessLogArchive purge`
  - `AccessLogExpire` *`number-of-days`*
- To remove access log files based on collective size, edit these directives:
  - `AccessLogArchive purge`
  - `AccessSizeLimit` *`number-of-megabytes`*
- To run a user exit, edit the `AccessLogArchive userexit` directive.

For details on these directives, refer to "AccessLogArchive - Remove existing access, agent, or referer log files or run a user exit" on page 493.

You can use a binary program or a shell script for your user exit.

Setting filters for the Access, Agent, and Referer logs

Note: You cannot set filters for the FRCA Access log.

For the Server Access log, you can set filters so that the Access, Agent, and Referer logs include only the information you are interested in.

To improve your ability to use the information included in the Access, Agent, and Referer log files, you can filter out extraneous information so that the log includes only information that is meaningful to you. You filter out information by excluding entries that match a particular pattern. We recommend you define these options on the **Access Log File Configuration** form, but you can edit the configuration file to include the appropriate directives for the filters you want to set. You can specify filters based on any of the following:

- URL
- IP address or host name
- Method
- MIME type
- Return code

Note: Keep in mind that information filtered out from the access log will not show up in any access report and will not be available for future use.

Here are some reasons for controlling what gets logged:

To reduce the size of the logs: You might be interested in reducing the number of entries in an access log to include only meaningful access requests. Access log files can grow rapidly, since by default they contain entries for all access requests for GIF images, HTML pages, and so on. You might want to configure your access

logs so that they include log entries for access requests to HTML pages, but not for the access requests for the GIF images that the HTML contains. For example, an HTML page might include several GIF images, which can cause the size of the access log to grow rapidly.

To collect information about external hits only: You might be interested only in who is accessing your server from outside your company. In this case, you would filter out access requests that originate from internal company IP addresses.

To gather information about who is accessing a particular Web site: To help you determine the size of the audience for a particular Web site, you might want to create an access log that shows only the hits to one URL.

Default: By default, everything is logged to the access log, unless you choose to filter out (exclude) something. From the **Access Log File Configuration** form, you can specify what you want to filter out from the access log. You do not need to fill in the entire form.

Scroll to the **Exclusions from the Access log** section of the form. Choose which of the following you want to base filtering on:
- Directories and files
- Host names or IP addresses
- Methods (GET, PUT, POST, DELETE)
- MIME types (images, text, applications, audio, video, multimedia, and other)
- Return code (success, redirection, client error, and server error)

If you want to filter based on directories and files or IP addresses and host names, you need to update the index list on the **Access Log File Configuration** form. You can insert or remove entries in the list to specify what you want filtered out. To exclude entries based on methods, MIME types, or return codes, click the boxes that describe what you want to filter out.

When you have finished specifying what you want to exclude on the **Access Log File Configuration** form, click **Submit** to have the filters take effect.

Directives:
- To filter out files or directories that match a particular pattern, edit the AccessLogExcludeURL directive.
- To filter out entries of a particular method, edit the AccessLogExcludeMethod directive.
- To filter out entries of a particular MIME type, edit the AccessLogExcludeMimeType directive.
- To filter out entries receiving a particular set of return codes, edit the AccessLogExcludeReturnCode directive.

Specifying options for the error logs

This section describes the following tasks:
- "Specifying the path for the Server Error and CGI Error logs"
- "Choosing log maintenance options for the Server Error and CGI Error logs" on page 195

Specifying the path for the Server Error and CGI Error logs: From the **Error Log File Configuration** form, you can specify the path and name of the directory

where you want to place the Server Error and CGI Error log files. As an alternative, you can specify this information manually by editing the directive listed below.

Directives: For the Server Error log, edit the ErrorLog directive. For more information, see "ErrorLog - Name the file where you want to log internal server errors" on page 503.

For the CGI error log, edit the CgiErrorLog directive. For more information, see "CgiErrorLog - Name the path for the CGI error log file" on page 503.

Choosing log maintenance options for the Server Error and CGI Error logs: You can choose whether you want to keep old logs, remove logs after they reach a certain age and/or a collective size, or run your own program at midnight each night to handle old logs. Note that the "collective size" is the collective size of all error logs only (not combined with CGI Error logs) or all CGI Error logs only (not combined with Server Error logs).

To reduce the space error logs require, you can specify that the logs be automatically removed, based on the age of the log and/or the collective size of the logs.

If you are interested in running your own backup program to store the logs, you can specify a user exit. In this case, you specify the path to your program and the parameters to pass to your program. The server appends to this information the path to the logs.

You can use a binary program or a shell script for your user exit.

Default: By default, all error and CGI error log files are kept at the path location you specify on the **Error Log File Configuration** form (or the ErrorLog directive).

We recommend you define these options on the **Error Log File Configuration** form, but you can edit the configuration file to include the appropriate directives. The settings you specify on the **Error Log File Configuration** form apply to CGI error logs, as well.

Directives: The directives you specify for error logs apply to CGI error logs, as well.

- To keep error log files, no directive is required. `ErrorLogArchive none` is the default.
- To remove error log files based on age, edit these directives:
  - `ErrorLogArchive purge`
  - `ErrorLogExpire` *`number-of-days`*
- To remove error log files based on collective size, edit these directives:
  - `ErrorLogArchive purge`
  - `ErrorSizeLimit` *`number-of-megabytes`*
- To run a user exit, edit the `ErrorLogArchive userexit` directive.

For details on these directives, refer to "ErrorLogArchive - Remove existing error or CGI error log files or run a user exit" on page 504.

Sample scenario for configuring log files

In the following example, you have just purchased and installed the HTTP Server. You want to set up your server to log access and error information in the following ways:

- You want the access and error logs to use a local time stamp and a common log file format.
- You have enabled dynamic caching using the Fast Response Cache Accelerator and want to log those requests in a separate access log file.
- You want the Server Access log files to be purged when they are more than 30 days old and/or when they reach a collective size of 25 megabytes. You do not want the following requests to be logged to the Server Access log:
  - Requests for GIF images
  - Requests from hosts with IP addresses that match 9.67.*.*
  - Redirection requests (requests that yield a return code between 300 and 399)
- You do not want the Server Error log to be purged.

You can specify these criteria by using the Configuration and Administration forms, or by updating the configuration file directives.

Forms

- Use the **Global Log File Configuration Settings** form to set the time and file format
- Use the **Fast Response Cache Accelerator** form to specify the path and file name for the FRCA Access log.
- Use the **Access Log File Configuration** form to:
  - Set the interval for removing old access logs
  - Set the collective size of 25 megabytes
  - Exclude the MIME type of images/GIF
  - Exclude requests from hosts with IP addresses in the pattern 9.67.*.*
  - Exclude requests that yield a return code between 300 and 399
  - Specify the path for the Server Access log file
- Use the **Error Log File Configuration** form to indicate that you want to keep the error log files.

Directives

For the above scenario, update the configuration file as follows:

```
FRCAAccessLog               /usr/lpp/internet/server_root/frca-log
LogFormat                   Common
LogTime                     LocalTime
NoLog                       9.67.*.*
AccessLogArchive            purge
ErrorLogArchive             none
AccessLogExpire             30
AccessLogSizeLimit          25
AccessLogExcludeURL         *.gif
AccessLogExcludeReturnCode  300
```

Tailoring the reports your server creates

Reporting options and considerations

IBM provides two reporting programs with the HTTP Server: HTLOGREP and Web Usage Mining. For information on these programs, see “Using the HTLOGREP or Web usage mining reporting programs” on page 197.

IBM also provides the option to use third-party reporting programs. See the following section for more information and instructions.

If you are using proxy server support, see "Proxy server considerations".

Configuring the Web server to use a third-party reporting program

You can specify a third-party reporting program using the Configuration and Administration Forms or by editing your server configuration file directly:

- To use the Configuration and Administration Forms:
  1. From the Front Page of your server, click **CONFIGURATION AND ADMINISTRATION FORMS**.
  2. Click **Logging and Reporting**.
  3. Click **Access Report File Configuration**. Refer to the online help for more information on configuration options.
  4. Restart the server to use the new settings.
- To edit the configuration file:
  1. Open the httpd.conf configuration file.
  2. Set the DoReporting directive `on` (default).
  3. Use the LoggingReportingProgram directive to specify the path and name of the third-party reporting program; use the LoggingReportingOptions directive to specify reporting program options.

     For information on these directives, see:
     - "DoReporting — Use logging and reporting options" on page 491
     - "LoggingReportingProgram — Specify the reporting program to be used" on page 509
     - "LoggingReportingProgramOptions — Specify reporting program options" on page 509

Proxy server considerations

If you have specified the CacheAccessLog directive or if you have indicated on the **Access Log File Configuration** form a path and file name for the proxy server's cache access log, your reports *will not* contain access requests for cached files. If you do not have a cache access log, access requests for a proxy server are logged in the access log and can be included in an access report.

Using the HTLOGREP or Web usage mining reporting programs

The following template-based reports are provided with your Web server:

- For HTLOGREP:
  - Host reports
  - Method reports
  - Code reports
  - URL reports
- For Web usage mining:
  - Daily mining statistics reports
  - Weekly mining statistics reports

You control what is included in reports by filtering out entries that match a particular pattern included in a report template. These options are defined using the Configuration and Administration Forms or by editing the configuration file. You can use the forms or the configuration file to specify filters based on any of the following:

- URL

- IP address or host name
- Method
- Return code

The contents of the report are governed by the following factors:
- The log file filters that were in effect when the log file was created
- The report filters that are in effect when the report is created

At report creation time, you control only the report filters that are currently in effect. You cannot include in the report entries that were filtered out from the log file.

You can specify report filters in two ways; you must decide which is easier in your situation.
- If you want to include in your report only a small percentage of the contents of the access log, it might be simplest to specify what to ***include***, rather than everything you want to exclude. Think of specifying one or a few include filters as a shortcut to specifying many exclude filters.
- If you want to include in your report most of the contents of the access log, it might be simplest to specify what to ***exclude***, rather than everything you want to include.

In some cases, you will find it simplest to specify both include and exclude filters. In this case, it is important to understand how include and exclude filters work together. The include filters are processed first. The report function searches the access log to find all entries that match any include filter patterns. If several include filters are specified, the filters act as OR Boolean expressions. In other words, entries that match at least one of the include filters are included.

The exclude filters are processed after all include filters have been processed. The exclude filters work only on the set of entries that have been already included by the include filters. For clarification, refer to the examples under "Sample scenarios for configuring reports" on page 200.

The include and exclude filters are specified on the **Access Log Report Template Creation** form or can be specified with the AccessReport directives.

Here are some reasons for controlling what gets reported:

To reduce the scope of the report: You might be interested in reducing the scope of the report so that it includes only a portion of what is contained in the log. You can even create several reports, each to gather different information from the same log. You might want to create your report template so that it includes log entries for access requests to HTML pages, but not for the access requests for the GIF images that the HTML contains.

To collect information about external hits only: You might be interested only in who is accessing your server from outside your company. In this case, you would filter out access requests that originate from internal company IP addresses.

To gather information about who is accessing a particular Web site: To help you determine the size of the audience for a particular Web site, you might want to create a report that shows only the hits to one URL.

To discover the top Web pages on your server: To help you determine the popularity of a particular Web site, you filter out everything in the report, except for the most visited Web pages.

Creating a report template

Before you create a report, you must modify or create a report template that outlines what you want the report to contain. To start configuring a report template, choose one of the following options from the **Access Log Report Templates** form:

- If you want to create a report that is very different from your existing reports, choose **Create a new template**.

  If you have never created a report, it might be easier to copy an existing template and edit the copy, rather than creating a new template. To use an existing report template as the basis of a new report, choose **Copy existing template**.
- To update an existing report template, choose **Edit existing template**.
- To delete an existing report template, choose **Delete existing template**.

When you choose **Create a new template** or **Edit existing template**, the **Access Log Report Template Creation** form appears.

On this form, you can specify some or all of the following:

- Basic settings, such as the report name, description, and the number of entries to include in the report (for example, the top 10)
- Entries from the access log that you want to exclude from the report (based on directories and file names, host names or IP addresses, methods, or return codes)
- Entries from the access log that you want to include (as a shortcut to many excludes)

Specifying entries to include is a shortcut to specifying many, many excludes. When you want to include only a few types of entries in the report, it is easier to specify what to include rather than excluding nearly everything. For example, if you want to include only access requests for a particular URL, you would include that URL, rather than excluding all the others.

The **Access Log Report Template Creation** form allows you to specify includes and excludes. It is important to understand how includes and excludes affect each other.

- In the index list, if nothing is listed with a Filter Action of “Include”, the entire log is included in the report, minus the entries that are excluded.
- If anything is listed with a Filter Action of “Include”, the report will contain only the information that is to be included, minus the entries that are excluded.

When you have finished filling in the form, click **Submit**.

The Web usage mining statistics are based on the report templates defined. To generate the report, you must either run webusage or wait until the current day’s log files are closed. This normally occurs at midnight when the current day’s logs are closed and the next day’s log files are started.

To initialize the report template files, you must either run htlogrep or wait until the current day’s log files are closed.

If you are using the **Access Log Report Templates** form, you see at the bottom of the form the field **Report root directory**. This field is filled in with a default directory. We recommend that you accept the default, rather than changing it. If you choose to change the default, you will need to create a new directory for the path you specify, give the directory the appropriate permissions and add a PASS statement to enable the server to honor requests to store reports in that directory.

Viewing reports

Reports are created dynamically through the use of a Jav™a applet accessed from your browser. Configuration directives are provided for the compression of log data, archiving of reports, and inclusion of old log data in reports. Refer to the Appendix B, "Configuration directives" on page 417 for details of these directives.

To see a report, from the Configuration and Administration Form page, choose **Access reports**. From there, select the following options:

- Report template
- One of the following filters:
  - URL
  - IP address or host name
  - Method
  - Return code
- Date range

By loading a template and then clicking on any report, you can display the report or print it from your Web browser. The report is created and displayed after you select the options.

By default, your server creates a report template that shows:

- The top 50 most frequently visited URLs on your server
- The top 50 most frequent visitors to your Web site

In addition, for each report template, the server creates a daily and weekly report of Web usage mining statistics. For a description of the Web usage mining statistics, see "Using the Web usage mining statistics reports" on page 203.

Sample scenarios for configuring reports

You have just purchased and installed the HTTP Server and you want to set up your server to automatically generate four different access log reports.

Sample report: Top 100 page hits

You are interested in knowing which Web pages on your server get the most attention. You decide to create a report that meets the following criteria:

- The name of the report is "Top100".
- The report description says "Top 100 page hits".
- Requests for GIF images are not included.

You can specify these criteria by using the Configuration and Administration forms, or by updating specific directives in the configuration file.

Forms:

1. From the Access Log Report Templates form, choose **Create a new template**. Select `before` in the **List it** field.
2. Change nothing in the **Root report directory** field.

3. Click **Submit**.
4. On the Access Log Report Template Creation form, for **Report name**, type `Top100`.
5. For **Report description**, type `Top 100 page hits`.
6. For **Report on top**, type `100`.

   Note: When you are using Web usage mining, the value for the AccessReportTopList directive must be an integer value up to 1000.
7. Scroll down the form.
8. Choose **Add** and **Exclude Directories/Files listed below**.
9. In the text box, type `*.GIF`.
10. Scroll down to the end of the form and click **Submit**.

Directives:

```
AccessReportTemplate  Top100   {
        AccessReportDescription     Top 100 page hits
        AccessReportTopList         100
        AccessReportExcludeURL      *.GIF
           }
```

Sample report: PUT requests to beta subdirectory

You are running a site that distributes beta-level software and are interested in knowing what is being written to the beta directory and who is requesting PUT access. You decide to create a report that meets the following criteria:

- The name of the report should be “BetaPuts”.
- The report description should say “PUT requests to beta subdirectory”.
- The report should include only requests for PUT access to the beta subdirectory, which is located at `/www/beta`.

You can specify these criteria by using the Configuration and Administration forms, or by updating specific directives in the configuration file.

Forms:

1. From the **Access Log Report Templates** form, choose **Create a new template**. Select `before` in the **List it** field.
2. Change nothing in the **Root report directory** field.
3. Click **Submit**.
4. On the **Access Log Report Template Creation** form, for **Report name**, type `BetaPuts`.
5. For **Report description**, type `PUT requests to beta subdirectory`.
6. Scroll down the form.
7. Choose **Add** and **Include Directories/Files listed below**.
8. In the text box, type `/www/beta/*`.
9. Scroll down the form.
10. Under **Exclude following Methods...** choose **GET**, **POST**, and **DELETE**.
11. Scroll down to the end of the form and click **Submit**.

Directives:

```
AccessReportTemplate  BetaPuts   {
        AccessReportDescription     PUT requests to beta subdirectory
        AccessReportIncludeURL      /www/beta/*
```

```
AccessReportExcludeMethod   GET
AccessReportExcludeMethod   POST
AccessReportExcludeMethod   DELETE
   }
```

Sample report: Accesses, excluding beta subdirectory and alpha7 requests

You are interested in knowing which files on your server are being accessed. However, you want to exclude beta programs, which have files located in the beta subdirectory. You also do not want to include any information on the "Alpha7" project, which has pages named `Alpha7*.*` in various subdirectories. You decide to create a report that meets the following criteria:

- The name of the report should be "NoBetaAlpha7".
- The report description should say "Accesses, except beta subdirectory and alpha7".
- The report should include all accesses, except those to the beta subdirectory at `/www/beta` or those files with the name `alpha7` anywhere on the server.

You can specify these criteria by using the Configuration and Administration forms, or by updating specific directives in the configuration file.

Forms:

1. From the **Access Log Report Templates** form, choose **Create a new template**. Select `before` in the **List it** field.
2. Change nothing in the **Root report directory** field.
3. Click **Submit**.
4. On the **Access Log Report Template Creation** form, for **Report name**, type `NoBetaAlpha7`.
5. For **Report description**, type `Accesses, except beta subdirectory and alpha7`.
6. Scroll down the form.
7. Choose **Add** and **Exclude Directories/Files listed below**.
8. In the text box, type:

   ```
   /www/beta/*
   alpha7*.*
   ```

9. Scroll down the form.
10. Scroll down to the end of the form and click **Submit**.

Directives:

```
AccessReportTemplate  NoBetaAlpha7   {
  AccessReportDescription   Accesses, excluding beta and alpha7 requests
  AccessReportExcludeURL    /www/beta/*
  AccessReportExcludeURL    alpha7*.*            }
```

Sample report: Accesses for department server and for the beta subdirectory except Alpha7*.* files

Your server is a department server and you want to know the access requests for that server. You also want to know access requests for the beta subdirectory, but you are not interested in knowing access requests for any Alpha7*.* files. You decide to create a report that meets the following criteria:

- The name of the report should be "DeptServer_Beta-NotAlpha7"
- The report description should say "Accesses for Department Server"

- The report should include all accesses to the IP address that represents the department server.
- The report should include all access requests to /www/beta/*.
- The report should exclude all access requests to Alpha7*.* files.

You can specify these criteria by using the Configuration and Administration forms, or by updating specific directives in the configuration file.

Forms:

1. From the **Access Log Report Templates** form, choose **Create a new template**. Select `before` in the **List it** field.
2. Change nothing in the **Root report directory** field.
3. Click **Submit**.
4. On the **Access Log Report Template Creation** form, for **Report name**, type `DeptServer_Beta-NotAlpha7`.
5. For **Report description**, type `Accesses for Department Server`.
6. Scroll down the form.
7. Choose **Add** and **Include Directories/Files listed below**.
8. In the text box, type:

   `/www/beta/*`

   To exclude the Alpha7*.* files, you will have to save the settings on this form, then edit the template. You cannot specify both include and exclude on the same form.
9. Choose **Add** and **Include host names listed below**.
10. In the text box, type 9.67.*.*
11. Scroll down to the end of the form and click **Submit**.
12. To exclude the Alpha7*.* files, choose **Edit existing template**.
13. Click **Submit**.
14. Scroll down the form.
15. Choose **Add** and **Exclude Directories/Files listed below**.
16. In the text box, type:

    `Alpha7*.*`
17. Click **Submit**.

Directives:

```
AccessReportTemplate  {
        AccessReportTemplate        DeptServer_Beta-NotAlpha7
        AccessReportDescription     Accesses for Department Server
        AccessReportIncludeURL      /www/beta/*
        AccessReportExcludeURL      Alpha7*.*
        AccessReportIncludeHostName 9.67.*.*
           }
```

Using the Web usage mining statistics reports

If you want to understand how your users navigate through your Web site, you can look at the Web usage mining statistics. They tell you the sequence of Web pages a user clicked through during a visit.

These reports can tell you where people enter and exit from your Web site and which Web pages as a group are visited most. You can see the browsing patterns

and identify user behavior, which in turn allows you to better organize your Web pages. The reports are generated automatically and are not tailorable except through the standard report templates.

There are three types of Web usage mining statistics reports:
- User-based
- Path-based
- Group-based

User-based statistics help you to understand how users move through your Web site. Each user session is recorded as the sequence of HTML links followed by a specific user. If a user remains idle for some pre-specified period of time, the next sequence of links is considered to be a new user session. The user-based statistic reports shows:
- Most frequently accessed pages organized by user count
- Most frequent IP addresses from which users come to visit your site organized by user count
- Distribution of user sessions both in duration and in number of pages accessed (bar charts are also supported when viewed with a Java-enabled browser)
- Most frequent external link (referer) to your site
- Most frequent page from which a user exits your site

Path-based statistics identify paths used to travel through your Web pages. Each user path is a sequence of HTML pages chosen to by a user and can reveal the user's actual browsing behavior. Path-based statistics tell you how the HTTP links embedded in a Web presentation are actually followed by users.

Group-based statistics give you the groups of pages most frequently visited during a user session, helping you to see which groups of pages are most popular. A user session can contain multiple paths; and the group of pages frequently visited in a session may not lie on the same path. By examining the path-based and group-based statistics, you can obtain valuable information to improve the organization and linkage of the Web presentation.

Daily reports are provided for all three types of statistics. In addition, a week-to-date user-based report is also provided. To view these reports, from the Configuration and Administration Form page, choose **Access Reports**. A list of the existing reports by template is displayed. One link exists for each template-based report. The reports reside on the reports directory as *template_name*_wumindex.MonYear.html. For example, Top50_wumindex.Sep1996.html contains the hyperlinks to all the daily and week-to-date reports for the month of September 1996. Here, Top50 is the name of a report template specified by the Webmaster for basic reporting.

For each report template, there is a corresponding set of user-, path- and group-based reports. The number of items reported in each statistic is specified in each report template. For example, on the Access Report Template Creation page at the **Report on Top** prompt, you can specify the number of top report items (25, 100, or all) you want to view.

Note: When you are using Web usage mining, the value for the AccessReportTopList directive must be an integer value up to 1000.

Notes:

1. HTLOGREP must be run before you can view Web usage mining reports. Either the Web server can run HTLOGREP automatically, or you can run it

manually. You can view the Web usage mining reports by using the Web server remote **Configuration and Administration** forms.

2. You must have set the AccessLog, AgentLog, and RefererLog directives to create the access, agent, and referer logs so the server can create the Web usage mining statistics reports. You also need to set the AccessReportTemplate directive.
3. The reports will be in .html format and will be stored in the path specified by the AccessReportRoot directive in the httpd.conf file.
4. Web usage mining reports can be viewed by selecting **Access reports** from the Configuration and Administration Form page and selecting the appropriate link.
5. When a year ends in the middle of the week, the week-to-date report will only contain statistics for the year that is ending. The part of the week that is in the new year will be in the first week-to-date report for the new year.
6. The information in a new week-to-date report will be included in a week-to-date report from a previous year when the following two situations occur:
   - The week-to-date reports are written to the directory pointed to by the AccessReportRoot configuration directive.
   - There is a week-to-date report from the previous year with the same suffix as the week-to-date report of the current year. Note that the suffix specifies the week.

   Example: 2001 is the previous year. 2002 is the current year. The directory on the AccessReportRoot directive is /usr/lpp/internet/server_root/pub/reports. The directory contains file Top_user.34 for week thirty-four of 2001. The server generates a week-to-date report for week thirty-four of 2002. The server then includes the information in the Top_user.34 file that was created in 2001.

Running Web usage mining from the command line

Before you can run the Web usage mining command, your sessions must be able to access the Web server's environment settings. This means that your user ID .profile settings must match the settings in the server's httpd.envvars file. To change environment variable settings in your .profile, use the export command, for example:

```
export NLSPATH=environment_variable_setting
```

Note: If you run the Web usage mining command and see the error message, `Error using message catalog`, this usually indicates that the NLSPATH setting in your .profile does not match the setting in the server's httpd.envvars file.

To run the Web usage mining command, type:

```
webusage -c full_path_of_configuration_file [-f httpd_ip_filename]
```

where

full_path_of_configuration_file
: is the full path of the active configuration file.

httpd_ip_filename
: is the filename of a file containing the IP address of the machine that generated the log files (for example, 9.1.2.3) and the fully qualified hostname of that machine (for example, myhost.XYACorp.com).

The required -c option provides the full path of the active configuration file.

Use the -f option when you want to run the Web usage mining tool on log files that were generated on another system. When using the -f option:

- The agent, httpd, and referer log files from the other system **must** be copied to the LOGS directory of the system where the Web usage mining tool will run.
- The IP address must appear in the first line of the IP file and the host name must appear on the second line. These two lines must be the only two lines in the IP file.

Web usage mining creates the appropriate reports in .html format and stores them in the root report directory.

Logs generated on another system **may** have different names than the logs generated on the system where the Web usage mining tool is running. In this case, you must change the appropriate configuration file directives for the Web usage minng tool to find the logs.

Web usage mining and multiple servers

Multiple servers can run on a single machine. Each server runs with a unique configuration file and listens on a unique port. For more information about running your server with multiple IP addresses or virtual hosts, see Chapter 17, “Running your server with multiple IP addresses or virtual hosts” on page 309.

The Web usage mining tool generates reports specific to each server. Each invocation of the Web usage mining tool has unique -c parameters and generates unique reports.

Logging information with System Management Facilities

With the System Management Facilities (SMF), you can request that configuration and performance data be recorded to SMF datasets. With this recorded configuration and performance data, you can monitor Web server health, throughput, and activity.

Configuration record data is taken from the server configuration file, httpd.conf, and is written after the server daemon is fully initialized. Performance record data is accumulated continuously and written at intervals defined in the httpd.conf file by the SMFRecordingInterval directive.

When multiple servers are managed by a workload manager, the totals written to SMF are for all servers under the workload manager.

You can choose to have either configuration record data or performance record data, both configuration and performance record data, or no data written to SMF datasets. You can also define how often SMF writes the continuously accumulated performance information to SMF datasets.

To select the type of information to be written to the SMF dataset, you can use the Global Log File Configuration Settings form in the Configuration and Administration forms or the SMF directive. The default setting for the SMF directive is `all`, which records both configuration and performance record data. For more information about the SMF directive, see “SMF - Specify the type of information that SMF records” on page 516.

To specify how frequently performance record information is written to SMF datasets, use the Global Log File Configuration Settings form in the Configuration and Administration forms or the SMFRecordingInterval directive. The default SMFRecordingInterval is `00:15`, which means performance record information is recorded every 15 minutes if the logging queue is full. If activity on the server is low, the logging queue fills slowly. In this case, the recording interval for performance record information may be longer than the specified interval. For more information about the SMFRecordingInterval directive, see "SMFRecordingInterval - Specify how often to record performance record information" on page 516.

For more information about the Configuration and Administration Forms see "Using the Configuration and Administration forms" on page 50.

Chapter 12. Customizing your Web site

Using the HTCounter program to display page count, date, time, and text on a Web page 209
Configuration instructions for the HTCounter program 209
Samples 210
Options 211
Common options 211
Counter option 212
Date and time option 212
Text to gif option 213
Server-side includes and support for text-based browsers 213
Using server-side includes to insert information into CGI programs and HTML documents. 214
Considerations for using server-side includes 214
Preparing to use server-side includes 214
Format for server-side includes 215
Syntax 215
Directives for server-side includes 215
config - controls file processing 215
echo - specify environment variables . . . 218
exec - specify CGI programs 218
cgi - specify CGI program URI 218
flastmod - display time and date document was changed 219
fsize - display file size 219
global - defines global variables 219
include - includes a document in output . . 219
set - sets variables to be echoed 220
Using server-side image maps to enable clickable images 221
Syntax 221
Examples 222

This chapter describes methods for customizing the appearance of your Web site. It includes information about displaying page count, date, time, and text on a Web page and using server-side includes to insert information into CGI programs and HTML documents.

Using the HTCounter program to display page count, date, time, and text on a Web page

This section explains how to use the HTCounter program to display the following information on a Web page:

Page count
The page counter is incremented each time the Web page is accessed, and the current value is displayed on the Web page.

Date and time
The current date and time are displayed on the Web page.

Text User-specified text is displayed on the Web page.

Configuration instructions for the HTCounter program

To use the HTCounter program:

1. Uncomment Service directives in the Web server configuration file for the functions you are using.

   In the following example, *install_path* is the root directory of your Web server installation. The default install path is /usr/lpp/internet.

   - Page count:

     ```
     # Service  /cgi-bin/apicounter*
     install_path/bin/htcounter.so:HTCounter*
     ```

   - Date and time:

     ```
     # Service  /cgi-bin/datetime*
     install_path/bin/htcounter.so:HTCounter*
     ```

   - Text:

```
# Service  /cgi-bin/text2gif*
install_path/bin/htcounter.so:HTCounter*
```

2. Create the counter file and initialize the counter.

   The Web administrator must create the counter file and initialize the counter to some value, for example, 0. This enables the Web administrator to control access to the Web site and to Web site resources used to display the requested information.

   The counter file can be located in either the default Counters directory (*server_root*/Counters) or in the directory you specify using the CounterDirectory directive.

   Notes:

   a. If you use the CounterDirectory directive, you must create a *CounterDirectory*/Fonts directory and create links or copy the *server_root*/Counters/Fonts directory into the *CounterDirectory*/Fonts directory.
   b. If you use the default Counters directory (*server_root*/Counters) but do not use the default ServerRoot directory (/usr/lpp/internet/server_root), copy the files in the *server_root*/Counters directory into your *server_root* directory.
   c. In some cases not having the contents of the *CounterDirectory*/Fonts or the *server_root*/Counters directories is not a problem; however, in other cases, the counter will not be displayed properly and error message IMW3728E, IMW3729E, or IMW3730E will be written to the error log.

   For information on sample HTML files, see “Samples”.

3. Insert lines in the Web page HTML file for the functions you are using.
   - Page count:

     ```
     <img src="/cgi-bin/apicounter/counter_name?[common_options]
     [counter_option]">
     ```

   - Date and time:

     ```
     <img src="/cgi-bin/datetime?[common_options]
     [date_and_time_option]">
     ```

   - Text:

     ```
     <img src="/cgi-bin/text2gif?[common_options]
     [text_to_gif_option]">
     ```

   Note: Place the information in the above examples on one line, even though it is shown here on two lines.

   For *counter_name*, enter the name of the counter file you created, for example, **cntfile.cnt**.

Samples

Sample HTML files are located in the *server_root* /pub directory, or you can go to the following URLs:

- To view the sample counter page, go to URL: http://*your.server.name*/counter1.html

  This page also includes an explanation of error messages that may be issued by the HTCounter program when you begin using the counter file.
- To view the sample color page, go to URL: http://*your.server.name*/counter2.html

For *your.server.name*, enter the fully qualified name of your host, for example **http://myserver.raleigh.ibm.com**.

Options

This section describes the options you can use for displaying the page count, date, time, and text on your Web page.

Notes:

1. Defaults are shown in **bold** letters.
2. Option names and values are **not** case-sensitive.
3. Use an ampersand (&) to separate options.

   For example, to display a page counter with a foreground color of blue and a background color of white, use the followingtag:

   ```
   <img src="/cgi-bin/apicounter/counter_name?FG=blue&BG=white">
   ```

4. The *RRGGBB* color option allows you to specify the color using a hexadecimal color code, where *RR*, *GG*, and *BB* are the hexadecimal digits that specify the Red, Green, and Blue values of the color. Examples of color values are:

   ```
   Black       000000
   Red         FF0000
   Orange      FFA500
   Green       00FF00
   Blue        0000FF
   Yellow      FFFF00
   White       FFFFFF
   ```

   For example, to display a page counter with a foreground color of yellow, you would use the following tag:

   ```
   <img src="/cgi-bin/apicounter/counter_name?FG=FFFF00">
   ```

   For colors demonstrated online, go to URL http://*your.server.name*/counter2.html or Lem Apperson's Color Index at http://www.infi.net/wwwimages/colorindex.html.

Common options

FG=*color*

where *color* specifies the foreground color, and can be:

Black
White
Red
Green
Blue
RRGGBB

BG=*color*

where *color* specifies the background color, and can be:

Transparent
Black
White
Red
Green
Blue
RRGGBB

BorderColor=*color*

where *color* specifies the border color, and can be:

Green
Black
White
Red

Blue
RRGGBB

BorderWidth=*width*
where *width* specifies the width of the border around the image, and can be:
0 No border (the default)
n The number *n* determines the thickness of the border.

BorderIndent=*highlighting*
where *highlighting* specifies highlighting for upper and right border edges (3D beveled effect), and can be:
In Upper and right border edges are shaded.
Out Lower and left border edges are lighter.

BorderIndentColor=*color*
where *color* specifies the color for the border edge (3D beveled effect), and can be:
BorderColor
Black
White
Red
Green
Blue
RRGGBB

BorderIndentWidth=*width*
where *width* specifies the width of the border edge (3D beveled effect), and can be:
0 No border edge (the default)
n The number *n* determines the thickness of the border edge.

FontName=*font*
where *font* specifies the font used, and can be:
Block1
LCD

FontSize=*size*
where *size* specifies the font size (width x height), and can be:
8x12
7x11
9x13
10x14

Counter option

Format=*format*
where *format* specifies the format for displaying the *counter_value*, and can be:
%%d No padding
%%*n*d Pad with blanks; width=*n*
%%0*n*d
Pad with zeros; width=*n*

Date and time option

Format=*strftime()-format*
Specifies the format for displaying the *date* and *time*:

- **Default**: http_time format
- *strftime()-format*

– Use %20 to represent a blank.
– For all other options, see Table 3 on page 216.

Timebase=*time*
Specifies the time used:
- **Local**
- GMT (Greenwich mean time)

Text to gif option

Text=*string*
Specifies the text string that will be converted to a gif. Use %20 to represent a blank.

Example:

The following URLs show an example of how you can display text and page count on a Web page:

```
 <img src="/cgi-bin/text2gif?BG=white&Text=
  This%20Web%20page%20has%20been%20accessed%20">
<img src="/cgi-bin/apicounter/counter_name?FG=red&BG=white">
<img src="/cgi-bin/text2gif?BG=white&Text=
  %20times%20since%20January%201,%201997.">
```

Note: In this example some image tags appear on two lines for printing purposes. Each image tag would appear on one line in your source file.

The URLs in this example display the following information on the Web page:

```
This Web page has been accessed n times since January 1, 1997.
```

In this example:
- The text is displayed in black (default) using a white background.
- The page count (*n*) is displayed in red on a white background.

Server-side includes and support for text-based browsers

Server-side includes can be used with the apicounter function to display counter values on text-based browsers.

Example:

```
<!--#exec cgi="/cgi-bin/apicounter/counter_name" -->
```

For *counter_name*, enter the name of the counter file to be displayed, for example, **sample.cnt**.

The server-side include returns a text string which can be displayed by either text or graphics-based browsers. HTML tags can also be used to format the result. The color and highlighting options described in "Common options" on page 211 cannot be used. However, the format option described in "Counter option" on page 212 can be used to pad the result with zeros or blanks.

Example:

```
<B><!--#exec cgi="/cgi-bin/apicounter/sample.cnt?Format=%%010d" --><B>
```

This example will display the counter file **sample.cnt** using the browser's bold font. The counter will be displayed with leading zeros.

Customizing your Web site

For more information and examples, go to URL http://*your.server.name*/counter1.html.

For a description of server-side includes and server configuration information, refer to "Using server-side includes to insert information into CGI programs and HTML documents".

Using server-side includes to insert information into CGI programs and HTML documents

Server-side includes allow you to insert information into CGI programs and HTML documents that the server sends to the client. This section describes the command format for using server-side includes and explains how to use the commands needed to make server-side includes work in your CGI programs and HTML documents.

Considerations for using server-side includes

Before using server-side includes on your server, note the following considerations:

- Performance

  Performance can be significantly impacted when the server is processing files while sending them.
- Security

  Letting ordinary users execute commands can be a security risk. Be very careful when deciding which directories you use server-side includes in and which directories you use the `exec` command in. You can minimize the security risk if you do not enable the `exec` command.
- File references

  You cannot reference files recursively. For example, if you are running file `sleepy.html` and the program finds `<--!#include file="sleepy.html" -->` the server doesn't detect the error and the server loops until the server abends.

  However, you can reference files within files. For example, file `sleepy.html` references file `smiley.html` and file `smiley.html` references `dopey.html`.
- Dynamic caching support

  Files containing server-side includes cannot be dynamically cached by the Fast Response Cache Accelerator. For more information on dynamic caching, see Chapter 10, "Customizing cache management with the Fast Response Cache Accelerator" on page 185.

Preparing to use server-side includes

To use server-side includes, you need to add the AddType directive to your configuration file. Two examples follow:

Examples:

```
AddType .shtml text/x-ssi-html ebcdic 1.0
AddType .htmls text/x-ssi-html ebcdic 1.0
```

Note: If you use file extensions other than .shtml or .htmls, you should check the AddType directive to see if that extension already exists. See the configuration file, appendix listing, or the MIME form for a list of existing AddType directives.

You can also use the imbeds directive to specify whether server-side includes can be used in HTML documents, CGI programs, or both. Examples of this directive follow:

Example:

```
imbeds on
```

Default

```
imbeds off
```

For more information about the imbeds directive, see "imbeds - Specify whether server-side includes will be dynamically imbedded" on page 451.

The server does not process your error files for imbeds, regardless of the file extensions or use of the imbeds directive.

Format for server-side includes

The current date, the size of a file, the last change of a file are examples of the kind of information that can be sent to the client. There are commands that need to be included in the HTML document comments. The commands have the following format:

Syntax

The following is the syntax format for enabling server-side includes on the server:

```
<!--#directive tag=value ... -->
<!--#directive tag="value" ... -->
```

The quotes around *value* are optional. They are required when there are imbedded spaces.

Directives for server-side includes

This section explains the directives that are accepted by the server for server-side includes.

config - controls file processing

Use this directive to control certain aspects of file processing. Valid tags are cmntmsg, errmsg, sizefmt, and timefmt.

cmntmsg - specify the message appended to the beginning of text: Use this tag to specify the message that gets appended to the beginnning of any text that follows a directive specification and comes before "-->".

Example:

```
<!--#config cmntmsg="[This a comment]" -->
<!--#echo var=" " extra text -->
```

Result: (Output from the `echo`) `<!--This is a comment extra text -->`

Default: `[the following was extra in the directive]`

errmsg - specify the message sent to the client: Use this directive to specify the message that gets sent to the client if an error occurs when a file is being processed. The message gets logged in the server's error log.

Example:

```
<!--#config errmsg="[An error occurred]" -->
```

Default: `"[an error occurred while processing this directive]"`

sizefmt - specify file size format: Use this directive to specify the format to be used when the file size is displayed. In the following examples, `bytes` is the value used for a formatted number of bytes. `abbrev` is used for displaying the number of kilobytes or megabytes.

Example 1:

```
<!--#config sizefmt = bytes -->
<!--#fsize file=foo.html -->
```

Result: `1024`

Example 2:

```
<!--#config sizefmt=abbrev -->
<!--#fsize file=foo.html -->
```

Result: `1K`

Default: `"abbrev"`

timefmt - specify date format: Use this directive to specify the format to be used when providing dates.

Example:

```
<!--#config timefmt="%T %D" -->
<!--#flastmod file=foo.html -->
```

Result: `"12:05:33 10/18/95"`

Default: `"%a, %d %b %Y %T %Z"`

The following strftime() formats are valid with the timefmt tag:

Table 3. Conversion Specifiers Used by strftime()

Specifier	Meaning
%%	Replace with %.
%a	Replace with the abbreviated weekday name.
%A	Replace with the full weekday name.
%b	Replace with the abbreviated month name.
%B	Replace with the full month name.
%c	Replace with the date and time.
%C	Replace with the century number (year divided by 100 and truncated).
%d	Replace with the day of the month (01-31).
%D	Insert the date as %m/%d/%y.
%e	Insert the month of the year as a decimal number (01-12). Under C POSIX only, it is a 2-character, right-justified, blank-filled field.
%E[cCxyY]	If the alternative date/time format is not available, the %E descriptors are mapped to their unextended counterparts. For example, %EC is mapped to %C.
%Ec	Replace with the alternative date and time representation.

Table 3. Conversion Specifiers Used by strftime() (continued)

Specifier	Meaning
%EC	Replace with the name of the base year (period) in the alternative representation.
%Ex	Replace with the alternative date representation.
%EX	Replace with the alternative time representation.
%Ey	Replace with the offset from %EC (year only) in the alternative representation.
%EY	Replace with the full alternative year representation.
%h	Replace with the abbreviated month name. This is the same as %b.
%H	Replace with hour (23-hour clock) as a decimal number (00-23).
%I	Replace with hour (12-hour clock) as a decimal number (00-12).
%j	Replace with the day of the year (001-366).
%m	Replace with the month (01-12)
%M	Replace with minute (00-59).
%n	Replace with a new line.
%O[deHlmMSUwWy]	If the alternative date/time format is not available, the %E descriptors are mapped to their unextended counterparts. For example, %Od is mapped to %d.
%Od	Replace with the day of the month, using the alternative numeric symbols, filled as needed with leading zeroes if there is any alternative symbol for zero, otherwise with leading spaces.
%Oe	Replace with the day of the month, using the alternative numeric symbols, filled as needed with leading spaces.
%OH	Replace with the hour (24 hour clock) using the alternative numeric symbols.
%OI	Replace with the (12 hour clock) using the alternative numeric symbols.
%Om	Replace with the month using the alternative numeric symbols.
%OM	Replace with the minutes using the alternative numeric symbols.
%OS	Replace with the seconds using the alternative numeric symbols.
%OU	Replace with the week number of the year (Sunday as the first day of the week, rules corresponding to %U) using the alternative numeric symbols.
%Ow	Replace with the weekday (Sunday=0) using the alternative numeric symbols.
%OW	Replace with the week number of the year (Monday as the first day of the week) using the alternative numeric symbols.
%Oy	Replace with the year (offset from %C) in the alternative representation and using the alternative numeric symbols.
%p	Replace with the local equivalent of AM or PM.
%r	Replace with the string equivalent to %I:%M:%S %p
%R	Replace with time in 24 hour notation (%H:%M)
%S	Replace with seconds (00-61).
%t	Replace with a tab.
%T	Replace with a string equivalent to %H:%M:%S.
%u	Replace with the weekday as a decimal number (1 to 7), with a 1 representing Monday.
%U	Replace with the week number of the year (00-53) where Sunday is the first day of the week.
%V	Replace with the week number of the year (01-53) where Monday is the first day of the week.

Table 3. Conversion Specifiers Used by strftime() (continued)

Specifier	Meaning
%w	Replace with the weekday (0-6) where Sunday is 0.
%W	Replace with the week number of the year (00-53) where Monday is the first day of the week.
%x	Replace with the appropriate date representation.
%X	Replace with the appropriate time representation.
%y	Replace with the year without the century.
%Y	Replace with the year with the current century.
%Z	Replace with the name of the time zone or no characters if the time zone is unknown.

The operating system configuration determines the full and abbreviated month names and years.

echo - specify environment variables

Use this directive to display the value for specified environment variables using the `var` tag. If a variable is not found, a **(None)** is displayed.For a list of the environment variables that you can use with server-side includes and display with the echo directive, see Appendix D, "Environment variables" on page 583.

Example:

```
<!--#echo var=SSI_DIR -->
```

Also, **echo** can display a value set by the **set** or **global** directives.

exec - specify CGI programs

Use this directive to include the output of a CGI program. Exec discards any HTTP headers CGI outputs EXCEPT for:

content-type
: determines whether to parse the body of the output for other Includes.

content-encoding
: determines if translation needs to be done (ebcdic/ascii).

last-modified
: replaces the current last modified header value if it is later.

cgi - specify CGI program URI

Use this directive to specify the URI of a virtual path to a CGI program.

Example 1:

```
<!--#exec cgi="/cgi-bin/program/path_info?query_string" -->
```

In the example, `program` is the cgi program to be executed, `path_info` are the parameters passed to the program as environment variables, and `query_string` are the parameters passed to the program as environment variables.

Example 2:

```
<!--#exec cgi="&path;&cgiprog;&pathinfo;&querystring;" -->
```

Example 2 shows the use of variables.

flastmod - display time and date document was changed

Use this directive to display the last time and date the document was changed. The formatting of this variable is defined using the `config timefmt` directive. The `file` and `virtual` tags are valid with this directive, and the meaning is the same as it is for the `include` directive.

Directive Formats:

```
<!--#flastmod file="/path/file" -->
<!--#flastmod virtual="/path/file" -->
```

Example:

```
<!--#flastmod file="FOO" extra text -->
```

Result: `12May96 <!-- This is extra text -->`

fsize - display file size

Use this directive to display the size of the specified file. The formatting of this variable is defined using the `config sizefmt` directive. The `file` and `virtual` tags are valid with this directive, and the meaning is the same as it is for the `include` directive.

Examples:

```
<!--#fsize file="/path/file" -->
<!--#fsize virtual="/path/file" -->
```

Result: `1K`

global - defines global variables

Use this directive to define global variables that can be echoed later by this file, or any included files.

Example:

```
<!--#global var=VariableName value="Some Value" -->
```

If you want to reference a parent document across the ″virtual″ boundary, you need to set a global variable DOCUMENT_URI. You also need to reference the global variable in the child document. The following is an example of the HTML coding you need to insert in the parent document:

Example:

```
<!--#global var="PARENT_URI" value=&DOCUMENT_URI; -->
```

The following is an example of the HTML coding you need to insert in the child document:

Example:

```
<!--#flastmod virtual=&PARENT_URI; -->
```

include - includes a document in output

Use this directive to include a document (the text from a document) in the output. The `file` and `virtual` tags are valid with this directive:

file - specify file name: Use this tag to specify the name of a file.

For **flastmod**, **fsize**, and **include**, **file** is assumed to be relative to SSI_ROOT if preceded by a '/'. Otherwise, it is relative to SSI_DIR. The file specified must exist either in SSI_ROOT or in one of its descendents.

Example:

```
<!--#include file="/path/file" -->
```

virtual - specify a document URI: Use this tag to specify the URI of a virtual path to a document.

For **flastmod**, **fsize**, and **include**, **virtual** is always passed through the server's mapping directives.

Example:

```
<!--#include virtual="/path/file" -->
```

set - sets variables to be echoed

Use this directive to set a variable that can be echoed later by only this file.

Example:

```
<!--#set var="Variable 2" value="AnotherValue" -->
```

Variables: Server-side includes also allow you to echo a variable already set. While defining a directive, you can echo a string in the middle of "value". For example:

```
<!--#include file="&filename;" -->
```

If an unrecognized variable is found, nothing gets displayed.

Server-side includes look for the variable, echo where the variable is found, and proceed with the function. You can have multiple variable references. When server-side includes encounter a variable reference INSIDE a server-side include directive, they attempt to resolve it on the SERVER side. The following example escapes the & so that server-side includes does not recognize it as a variable. In the second line, the variable "&index;" is a server-side variable and is used to construct the variable name "var1". The variable ê is a client side variable, so the & is escaped to create the value ":frêd" or "fred" with a circumflex over the e.

```
<!--#set var="index" value="1" -->
<!--#set var+"var&index;" value+"fr\&ecirc;d" -->
<!--#echo var="var1" -->
```

The following characters that can be escaped. Note that escaped variables are preceded with a backslash (\).

```
\a     Alert(bell)
\b     Backspace
\f     Form feed (new page)
\n     New line
\r     Carriage return
\t     Horizontal tab
\v     Vertical tab
\'     Single quote mark
\"     Double quote mark
\?     Question mark
\\     Backslash
\-     Hyphen
\.     Period
\&     Ampersand
```

Using server-side image maps to enable clickable images

Use the htimage service to process clickable image maps. It allows defined regions within an image map to be associated with specific URLs. When users click on a defined region, the htimage service redirects the server to the URL associated with the region. When the server returns the URL associated with the region, it issues a 302 return code (Moved Temporarily) and a Location header containing the URL.

Note: The URL specified in the Location header must be fully qualified, or absolute. The URL should include the protocol, host, and request path. For example:

```
http://hostnamefilename.html
```

The HTTP Server assumes, if the URL in the map file is not absolute, that the URL is local, and serves it directly. While this saves some network overhead, the browser thinks the original request was satisfied and continues to use the original request as the base for future requests.

When the URL is not fully qualified and the server finds the file locally, the HTTP Server issues a Content-Location header which some browsers use to make future requests.

Many browsers understand the HTML tag, <BASE HREF= >, which can be used to specify the base URL (for example, `http://hostname`). When this tag is used, URLs that are not fully qualified are evaluated relative to this base URL and are correctly found.

The service is used in conjunction with a map file. Map files are text files that define regions within a graphics file by their x,y coordinates and map them to the various URLs.

Note: Currently the universal image file accepted by all browsers and servers is the GIF format. GIF is an 8-bit 256 color image file.

You cannot use htimage from the command line. You include it as part of an anchor tag within an HTML document, and it is called when the server processes that document.

The htimage service is enabled by specifying the following in the server configuration file:

```
service /cgi-bin/htimage* INTERNAL:HTImage*
service /cgi-bin/imagemap* INTERNAL:HTImage*
```

Syntax

Since htimage can only be called from an anchor tag within an HTML document, the syntax is shown as HTML markup.

```
<a href="/cgi-bin/htimage/mapfile.txt">
<img src="/icons/image.gif" ISMAP></a>
```

<a href="/cgi-bin/htimage/*mapfile.txt***">**
The href attribute of the anchor (a) tag specifies the URL of the htimage command followed by the URL of the map file.

This syntax description assumes that the sample configuration file is being used. The sample configuration file contains Service and Exec directive that maps requests beginning with /cgi-bin/ to the directory that contains the htimage service.

The server uses everything following /htimage/ as the URL of the map file. If the server is using the sample configuration and the URL contains only a file name, the server would look for the file in the document root directory.

<img src="/icons/*image.gif***" ISMAP></A>**
The src attribute of the image (img) tag specifies the URL of the file that contains the graphic you want to use as an image map.

The ISMAP attribute indicates that the graphic is an image map.

Examples

Following is a description of the map files that must be used with the htimage service. A map file is an EBCDIC text file.

Note: Each line of the file is in the following format:

```
region-identifier [region-definition]  URL
```

region-identifier
A keyword that identifies the type of region being defined. Valid values are:

- rectangle
- circle
- polygon
- default

region-definition
A set of numbers that defines a particular region of the graphic. The format of the region definition is different for each type of region.

Coordinates within parentheses identify a point relative to the top left corner of the image. The first number is the number of pixels to the right of the top left corner. The second number is the number of pixels down from the top left corner. There are several shareware programs available that can help you easily identify the coordinates of particular points within a graphic file.

```
default   URL
rectangle (x1,y1) (x2,xy) URL
circle    (x,y)   r       URL
polygon   (x1,y1) (x2,y2) (x3,y3) URL
```

- The `default` keyword does not define a region. The keyword is followed by a URL to link to when the client clicks on a portion of the image map that is not covered by any of the other region definitions.
- For `rectangle`, the first point is the upper left corner of the rectangle. The second point is the lower right corner. In other words, define any two diagonally opposite corners having coordinates (x1,y1) and (x2,y2).
- For `circle`, the point is the center of the circle. The single number following the point is the radius of the circle as measured in pixels.
- For `polygon`, up to 100 points can be defined. The shape is formed by connecting the points in the order they are given. The last point is connected to the first.

For example, you might use the following HTML in a document:

```
<a href="/cgi-bin/htimage/mapit.txt">
<img src="/icons/mapimage.gif" ISMAP></a>
```

The above example calls the htimage command with a map file named mapit.txt. The mapit.txt file would define regions of the mapimage.gif graphic file. Because no path is specified for mapit.txt, the server would look for it on the document root directory. Following is an example of what mapit.txt might look like.

```
default                                  http://brimstone/cgi-bin/go_home
rectangle (50, 100) (200,200)            http://brimstone/cgi-bin/go_to_it
circle    (100,300) 50                   http://brimstone/pub/example.html
poly      (450,350) (450,500) (150,500) http://brimstone/pub/triangle.html
```

Note: There is an HTML tag, BASE, which can be used to specify the base URL (for example, `http://hostname`) so that relative URLs are evaluated relative to this base URL and are correctly found.

One example of a shareware program for determining the x,y coordinates is mapedit. You can obtain mapedit from the following URL:

http://www.boutell.com/mapedit/

Please note that output from mapedit is in NCSA format. It is different from the CERN examples shown.

The same examples in NCSA format require anchor tags within an HTML document as follows:

```
<a href="/cgi-bin/imagemap/mapit.txt">
<img src="/icons/mapimage.gif" ISMAP></a>
```

The map file will be an EBCDIC text file with the following format:

```
default http://brimstone/cgi-bin/go_home
rect    http://brimstone/cgi-bin/go_to_it  50, 100  200,200
circle  http://brimstone/pub/example.html  100,300  100,350
poly    http://brimstone/pub/triangle.html 450,350  450,500  150,500
```

URL

The fully qualified or absolute URL, including the protocol, hostname and filename, is required. If the URL in the map file is not absolute, the HTTP Server assumes that the URL is local and serves it directly.

Chapter 13. Managing your Web server

Modes of Operation for the Web server. 225
Standalone Server mode. 226
Scalable Server mode. 226
Single IMWHTTPD Queue Manager job/process 226
(Zero to n) IMWHTTPD Queue Server jobs/processes 226
Advantages 227
Disadvantages 227
Running Multiple Servers 227
Workload Management Enablement for the Web server 228
Workload management overview. 228
Configuring workload management to support the Web server 228
How workload management panels are interpreted 233
Configuring your environment for workload management 234
Configuring your application environment name for workload management 234
Example 1: Setting up a single scalable server on your system. 235
Summary of tasks 235
Step 1: Modify the Web server sample Workload Management policy and then load it into a Workload Management couple data set. 235
Step 2: Define the queue manager and queue server procedures. 236
Step 3: Set up the Web server configuration file to support a scalable server. 238
Step 4: Verify that your scalable server can successfully serve a request. 238
Example 2: Setting up multiple scalable servers on your system. 241
Summary of tasks 242
Step 1: Modify the Web server's sample WLM policy and then load it into a WLM couple data set.. 242
Step 2: Define the queue manager and queue server procedures. 243
Step 3: Set up the Web server configuration file to support a scalable server. 245
Step 4: Verify that your scalable server can successfully serve a request. 246
Server Activity Monitor 249
Enabling the Server Activity Monitor 249
Accessing statistics and usage information. . . 249
Web server activity statistics 250
Thread counts 250
Request statistics 251
Throughput statistics 251
Connection counts. 252
Response times (in seconds) 252
Network activity statistics 252
Access log entries 253
Simple Network Management Protocol 253
SNMP commands and protocol 253
Object IDs and variable names for the HTTP Server MIB 254
Defining a management information base file so that you can do queries with variable names . . 269
Simple Network Management Protocol examples: running with a single Web server and a single Simple Network Management Protocol agent 270
Example 1: Using an empty /etc/mibs.data file 270
Example 2: Using an updated /etc/mibs.data file 271
Example 3: Employing user defined variables in the /etc/mibs.data file 272
Simple Network Management Protocol examples: running multiple Web servers, each with a unique Simple Network Management Protocol agent 273
Server 1 examples 274
Server 2 examples 275
Creating an e-mail address to receive Simple Network Management Protocol problem reports. 276
Providing a security password for Simple Network Management Protocol 276
Enabling and disabling SNMP support 277
Turning the SNMP support on and off from the IMWHTTPD program 277
System Management Facilities. 277
Turning SMF support on and off from the IMWHTTPD program 278
Turning SMF on and off with the z/OS operator console MODIFY command 278
Controlling the logging of information by SMF 278
SMF record formats 279
Self-defining section of the SMF record header. 279
SMF configuration record data area (record type 103, subtype 01). 280
SMF performance record data area (record type 103, subtype 02). 281
z/OS console commands 284

Modes of Operation for the Web server

There are three execution modes for the Web server: Standalone Server mode, Scalable Server mode, and running multiple servers. This section lists the characteristics of these execution modes.

It is important to understand these execution modes. These server execution modes are discussed here so you will have an overview of the characteristics of these modes and where workload management, simple network managment protocol and system management facilities work.

Standalone Server mode

The following is a list of the characteristics of a standalone server:

- Single IMWHTTPD Web daemon job/process
- Console interface
- http: and https: accept loops
- Restart (SIGHUP) and shutdown (SIGTERM) handlers
- Proxy cache manager
- All requests are handled
- PID and log files
- SMF records, SNMP subagent

Scalable Server mode

When running in Scalable Server mode, the Web server works in conjunction with z/OS Workload Management (WLM).

The following is a list of the characteristics of a scalable server subsystem:

Single IMWHTTPD Queue Manager job/process

- Console interface
- http and https: sockets
- Restart (SIGHUP) and shutdown (SIGTERM) handlers
- PID and log files
- Proxy cache manager
- SMF records, SNMP subagent
- Creates a unique subsystem instance (-SN parameter)
- Handles unrouted (non-ApplEnv) requests
- Creates shared memory configuration/parameters

(Zero to n) IMWHTTPD Queue Server jobs/processes

- Joins a unique subsystem instance (-SN parameter)
- Uses shared memory configuration/parameters
- No console interface
- No accept loops
- No restart handler
- PID and log file (per IMWHTTPD)
- No proxy cache manager
- No SMF records, and no SNMP subagent. However, counters are accumulated and can be accessed through the Web server Queue Manager.
- Handles requests routed to one ApplEnv (-AE parameter) and subsequent unrouted requests on persistent connections
- Auto-started (by WLM based on policies) or pre-started (after Queue Manager for -SN)

Advantages

- WLM monitors and controls servers automatically to meet server demand.
- WLM prioritizes Web server work along with other MVS work.
- You can make changes to WLM service definitions and refresh them without incurring an outage.
- You can do better classification of work and better isolation of work that is based on your business needs.
- Scalable server mode allows for a large number of concurrent persistent connections.

Disadvantages

- Scalable server mode is more complex to set up.
- There is a slightly increased path length for requests because of workload management and cross address space communication.
- There is increased extended system queue area (ESQA) utilization because communication between a queue manager and a queue server is via shared memory.
- There can be storage utilization issues because of multiple address spaces.

Running Multiple Servers

Running multiple servers allows you to have multiple instances of the Web server running simultaneously on different ports. To do this, you must have separate configuration files and ports.

Some resources cannot be shared across server instances:

- Port number

  If your server is running without BindSpecific, port numbers MUST be unique. If your server is running with BindSpecific, port numbers must be unique within the given IP address.
- PID file

  The absolute PidFile must be unique. If the PidFile is a relative name, then ServerRoot and PidFile are combined to create an absolute name.
- Log files

  Absolute LogFile names must be unique. If specified as relative names, then ServerRoot and LogFile are considered.
- Proxy cache manager (garbage collection)

  Servers can share a proxy cache, but only one server should do proxy cache management for a given proxy cache.
- Scalable server subsystem instance name

  Each Queue Manager must have a unique SubSystem Name (-SN). Each Queue Server must identify which SubSystem Instance it is joining (-SN).
- SNMP MIB

  If you are running multiple servers, you can either enable SNMP on only one server or install one SNMP agent for each SNMP-enabled server.

Workload Management Enablement for the Web server

Workload Management (WLM) is a component of the z/OS operating system. WLM provides the functions to define, implement, and monitor system performance against business goals. WLM uses policies you define to match resources to workloads.

The Web server allows you to use workload management to establish policies to manage workload. The scalable server subsystem works in conjunction with workload management. z/OS systems typically run a wide variety of applications with conflicting resource requirements and priorities. Workload management modifies the environment to balance workload and improve system performance.

Workload management overview

Workload management (WLM) allows you to establish policies for managing the server environment. Based on these policies, workload management modifies the environment to achieve the best overall results which match individual needs. You can divide incoming requests into Application Environments. Each application environment is treated as a workload management queue of transactions. HTTP requests are routed to workload management transaction queues based on the ApplEnv directives specified in your server configuration file.

Workload management may change scheduling priority of individual threads. WLM may also change the number of processes working on an application environment.

The description that follows tells you how to define application environments using ISPF panels. For more information about WLM panels, see *z/OS MVS Planning: Workload Management*.

Configuring workload management to support the Web server

You use the WLM administrative application to set up the application environment and classification rules. The workload management application panels are automatically installed with your z/OS system. Start the WLM ISPF administrator dialogs by selecting WLM from the ISPF Primary Option Menu.

The following screen is the “Choose Service Definition” panel. This panel allows you to:

- Read an existing policy (option 1)
- Read the policy already installed on your coupled data set, if it applies (option 2)
- Create a new policy (option 3)

A sample policy has been included with the Web server. The data set name is called ’IMW.SIMWTBL1’.

To use the sample policy, select option 1. If you want to create a new policy, select option 3.

Note: The following information describes how to modify or use the sample that is provided with the Web server.

```
File   Help
------------------------------------------------------------
Command ===> ____________________________________________


              ------------------------------------------
              |      Choose Service Definition         |
              |                                        |
              | Select one of the following options.   |
              | 1   1.  Read saved definition          |
              |     2.  Extract definition from WLM    |
              |         couple data set                |
              |     3.  Create new definition          |
              |                                        |
              | F1=Help      F2=Split     F5=KeysHelp   |
              | F9=Swap     F12=Cancel                 |
              ------------------------------------------
                        ENTER to continue



F1=Help    F2=Split    F3=Exit    F9=Swap    F10=Menu Bar    F12=Cancel
```

After selecting option 1, the following panel is displayed. Enter the data set name that contains WLM policy information in the appropriate field and press enter to continue. This causes WLM to read the sample and display the WLM configuration panels.

```
File   Help
------------------------------------------------------------
Command ===> ____________________________________________


              ------------------------------------------
              |      Choose Service Definition         |
              |                                        |
              | Select one of the following options.   |
              | 1   1.  Read saved definition          |
 ---------------------------------------------------------------------
 |                      Read saved definition                        |
 |                                                                   |
 |                                                                   |
 | Data set name . . . 'IMW.SIMWTBL1'                        |
 |                                                                   |
 | Type a ? for a list of previously defined service definition data |
 | sets.                                                             |
 |                                                                   |
 |                                                                   |
 | F1=Help   F2=Split   F5=KeysHelp   F9=Swap   F12=Cancel           |
 ---------------------------------------------------------------------

F1=Help    F2=Split    F3=Exit    F9=Swap    F10=Menu Bar    F12=Cancel
```

The following screen is the “Definition Menu” panel that allows you to set up WLM to your individual needs. From this screen you can select option 6 or 9 to change various WLM parameters to meet your needs.

Managing the Web server

By specifying option **6** on the Definition Menu, you can view a list of subsystem types defined to WLM. IWEB is reserved to WLM for the Web server.

```
File   Utilities  Notes   Options  Help
------------------------------------------------------------
WLM                 Definition Menu              LEVEL003
Command ===> ____________________________________________
Definition data set . . . . :none

Definition name . . . . . . WEB_GOAL  (Required)
Description . . . . . . . . Web Service Definition

Select one of the
following options . . . . . 6_  1. Policies
                                2. Workloads
                                3. Resource Groups
                                4. Service Classes
                                5. Classification Groups
                                6. Classification Rules
                                7. Report Classes
                                8. Service Coefficients/Options
                                9. Application Environments





F1=Help    F2=Split    F3=Exit    F9=Swap    F10=Menu Bar    F12=Cancel
```

The following screen is the Subsystem Type Selection List for Rules. One or more applications, in addition to the server, can be displayed. To modify the server (IWEB) policy, select option 3 and press enter to continue.

```
Subsystem-Type  View  Notes  Options  Help
-------------------------------------------------------------------
              Subsystem Type Selection List for Rules    Row 1 to 5
Command ===> ___________________________________________


Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar


                                                 -----Class-----
Action  Type      Description                    Service    Report
  ___   ICT       OpenMVS ICT                    TSOCLASS
  _3_   IWEB      WEB Server Test Subsystem      FAST
  ___   JES                                      DISCRETN   BAT00
  ___   STC       Started Tasks
  ___   TSO       TSO users                      TSOCLASS

******************* Bottom of data ********************************




F1=Help    F2=Split    F3=Exit    F9=Swap    F10=Menu Bar    F12=Cancel
```

Use the following screen, "Modify Rules for the Subsystem Type" panel, to define classification rules. These classes are defined under option 4 of the definition panel.

```
Subsystem-Type  Xref  Notes  Options  Help
-----------------------------------------------------------------------
               Modify Rules for the Subsystem Type    Row 1 to 5 of 5
Command===> ____________________________________SCROLL ===> PAG

Subsystem Type . : IWEB     Fold qualifier name?  Y  (Y or N)
Description  . . . WEB Server Test Subsystem

Action codes:  A=After. . C=Copy         M=Move     I=Insert rule
               B=Before   D=Delete row   R=Repeat   IS=Insert Sub-rule
            -------Qualifier ----------      -------Class-------
Action    Type      Name     Start              Service      Report
                                 DEFAULTS:  FAST        ___________
__ 1  TN        POST     _____              IBMUSER    ___________
__ 1  TC        CGI      _____              SLOW       ___________
__ 1  SPM       009.009. _____              MEDIUM     ___________
__ 2    SPM       129.*  _____              ______     ___________
__ 1  UI        WEBSRV   _____              FASTPER3   ___________
```

When you are done with this screen, select PF12 to return to the previous screen.

After reading in the sample, you may want to add, delete, or modify some of the Application Environments based on your individual needs. To do this, specify **9** to display a list of the currently defined Application Environments.

```
File   Utilities  Notes   Options  Help
-------------------------------------------------------------
WLM                 Definition Menu               LEVEL003
Command ===> ___________________________________________
Definition data set . . . . :none

Definition name . . . . . . WEB_GOAL  (Required)
Description . . . . . . . . Web Service Definition

Select one of the
following options . . . . . 9_  1. Policies
                                2. Workloads
                                3. Resource Groups
                                4. Service Classes
                                5. Classification Groups
                                6. Classification Rules
                                7. Report Classes
                                8. Service Coefficients/Options
                                9. Application Environments





F1=Help    F2=Split    F3=Exit    F9=Swap    F10=Menu Bar    F12=Cancel
```

The following screen is the "Application Environment Selection List". The sample includes three application environments. Each environment is configured to

manage a specific function. WEBCGI is a sample application environment to process CGI command requests; WEBCICS is a sample application environment for CICS processing; and WEBHTML is a sample application environment for HTML processing. To create new application environments, select option 1. To modify an existing application environment, select option 3.

```
Application-Environment  Notes  Options  Help
-----------------------------------------------------------------
              Application Environment Selection List     Row 1 to 3
Command ===> ____________________________________________


Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar

Action   Application Environment Name    Description
  ___     WEBCGI                         HTTP Test Environment
  ___     WEBCICS                        HTTP Test Environment
  _3_     WEBHTML                        HTTP Test Environment
**********************Bottom of data ******************************









F1=Help    F2=Split    F3=Exit    F9=Swap    F10=Menu Bar    F12=Cancel
```

You must define at least one application environment in order for WLM to start server address spaces. You may define more than one application environment if there are functions that have widely varying resource requirements, and the requests for those functions cannot be divided into separate service classes. Workload management maintains separate address spaces for each combination of application environment and service class that appears in the workload.

The following screen is the "Modify an Application Environment" panel and shows you how to create application environments. The variable, &IWMSSNM, is resolved to the subsystem name by the server PROC.

```
Application-Environment  Notes   Options   Help
-----------------------------------------------------------------
               Modify an Application Environment
Command===> ____________________________________________________

Application Environment . . . WEBHTML_________________Required
Description . . . . . . . . . HTTP test environment___
Subsystem Type  . . . . . . . IWEB    Required
Procedure Name  . . . . . . . IMWIWM
Start Parameters  . . . . . . IWMSN=&IWMSSNM,IWMAE=WEBHTML
                              ______________________________________
                              ________________________


Limit on starting server address spaces for a subsystem instance:
1   1.  No limit
    2.  Single address space per system
    3.  Single address space per sysplex
```

After modifying or creating new application environments, press PF12 to return to the main WLM panel. To complete the workload management configuration process, perform the following steps:

1. Select file, then save.
2. Install the WLM definition. Go to the first panel, put the cursor under UTILITIES, and choose INSTALL DEFINITION. This installs your changes to the coupled data set.
3. Activate the service policy. Go to the first panel, put the cursor under UTILITIES, and choose ACTIVE SERVICE POLICY.
4. Exit the WLM application.
5. Turn WLM on. Enter F WLM,MODE=GOAL on the operator console.
6. Start the server using the -SN parameter. The parameter should be added to the start-up PROC.

Information about configuring RACF to support WLM, starting the queue server with WLM, and common WLM problems can be found in Chapter 3, “Installing your secure server” on page 19.

How workload management panels are interpreted

You can start the queue manager by specifying a subsystem name as WLMUT1 (-SN WLMUT1). The substitution occurs as follows:

1. When WLM attempts to start the queue server, WLM sees &IWMSSNM in the start parameters and substitutes WLMUT1 for it before issuing the z/OS start command. The command looks like this:

   ```
   S IMWIWM,IWMSN=WLMUT1,IWMAE=WEBHTML
   ```

2. The JCL for IMWIWM goes through the converter and interpreter (C/I) before you get control in the new address space. C/I processing sees all the &xxxxx symbols and tries to resolve them to values. &SN, &AE, and &QQ are easy to resolve because they are SET in the PROC. &IWMSN and &IWMAE must be resolved from the start command. Specifying IWMSN=,IWMAE= on the PROC statement tells z/OS that those symbols are inputs from the start command.

z/OS goes to the start command, gets the values, WLMUT1 and WEBHTML respectively, and substitutes them into the JCL.

3. After C/I is done and IMWEBSRV gets control, ICSPARM will have all the correct values substituted.

```
ICSPARM='-SN WLMUT1 -AE WEBHTML'
```

For more detailed explanations of how to use the WLM application panels, see *z/OS MVS Planning: Workload Management*.

Configuring your environment for workload management

To configure your environment for workload management, follow the directions in Chapter 3, "Installing your secure server" on page 19. This step instructs you on the following:

- Putting the IMWEBSRV and IMWIWM procedures in a site procedure library, such as SYS1.PROCLIB. IMWEBSRV is a procedure for the standalone server or the WLM Queue Manager. IMWIWM is the procedure that can be started automatically by WLM for the Queue Server.
- Defining the IMWIWM procedure to the started task table.
- Assigning the same user ID to IMWIWM as you do to the server.

Notes:

1. A RACF profile must be defined for the subsystem IWEB.
2. Only -SN and -AE can be passed on ICSPARM's in the IMWIWM procedure. Any other parameters cause the procedure to fail.
3. IMWIWM invokes IMWEBSRV by default. If you are using procedures other than these defaults, make sure the procedure that starts the Queue Server invokes the procedure that starts the Queue Manager. This insures that the Queue Manager and Queue Servers are invoked using the same Language Environment parameters.

Configuring your application environment name for workload management

To configure your application environment name for workload management, follow the directions in Appendix B, "Configuration directives" on page 417. This section instructs you on how to set up the following workload management directives:

- ApplEnv
- ApplEnvConfig
- PluginDefault
- PluginExclude
- PluginInclude
- ApplEnvMin
- ApplEnvMax
- ApplEnvPrestart

You can use the Workload Management Configuration and Administration Forms to set up your application environment name.

Example 1: Setting up a single scalable server on your system

You can run either a single scalable server or simultaneously run multiple scalable servers on your system. This example shows how to implement a single scalable server.

In this example you set up a scalable server so that the queue manager routes HTML requests to one queue server and GIF requests to another queue server.

Before you begin:

- Ensure that you have Workload Management (WLM) installed and running on your system. For information on Workload Management, see *z/OS MVS Planning: Workload Management* and *z/OS Programming: Workload Management Services*. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.
- Ensure that the user IDs issuing RACF commands, issuing console commands, and accessing WLM ISPF administration dialogs have the proper authority.

Summary of tasks

In this example, you perform the following main tasks:

1. Modify the Web server sample Workload Management (WLM) policy and then load it into a WLM couple data set.
2. Define the queue manager and queue server procedures.
3. Set up the Web server configuration file to support a scalable server.
4. Verify that the scalable server can successfully serve a request.

Step 1: Modify the Web server sample Workload Management policy and then load it into a Workload Management couple data set.

1. Permit the Web server to use WLM and the WLM administration policies by issuing the following RACF commands.

   ```
   RDEFINE FACILITY MVSADMIN.WLM.POLICY UACC(NONE) NOTIFY(WEBSRV)
   RDEFINE FACILITY BPX.WLMSERVER UACC(NONE) NOTIFY(WEBSRV)
   PERMIT MVSADMIN.WLM.POLICY CLASS(FACILITY) ID(WEBSRV) ACCESS(UPDATE)
   PERMIT BPX.WLMSERVER CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

2. Define a workload in the WLM ISPF administrator dialogs to manage your Web server work. This example starts with the sample WLM policy shipped with the Web Server to define the workload. The data set name containing the sample policy is called IMW.SIMWTBL1. The sample contains two application environments: WEBHTML and WEBCGI. You create a third application environment called WEBDFLT. You specify all three of these application environments on ApplEnv directives in the Web server configuration file. Define any application environments that you specify on ApplEnv directives under Application Environments in the WLM administrator dialogs. To view the sample policy in the WLM administrative dialogs, see "Configuring workload management to support the Web server" on page 228.

   To modify the Web server sample WLM policy and then activate it, complete the following steps:

   a. Start the WLM ISPF administrator dialogs by selecting **WLM** from the **ISPF Primary Option** menu.
   b. Press Enter to continue.
   c. Enter **1** to read a saved definition and then press Enter.

d. Type in the data set name of the saved definition, **'IMW.SIMWTBL1'** and then press Enter.

e. Enter **9** to select the Application Environment option on the **Definition** menu and then press Enter.

f. Enter **1** under Action on the **Application Environment Selection** list, to create a new application environment name,and then press Enter.

g. Complete the fields on the **Create an Application Environment** panel and then press **PF3** to save the results:

- Enter the Application Environment: **WEBDFLT**.
- Enter the Description: **handles remaining requests**.
- Enter the Subsystem Type: **IWEB**.
- Enter the Procedure Name: **IMWIWM**.
- Enter the Start Parameters: **IWMSN=&IWMSSNM,IWMAE=&WEBDFLT**.
- Keep the default of **1**, No Limit, for the limit on starting server address spaces for a subsystem instance.

h. Press **PF12** to return to the **Definition** menu.

i. Put the cursor under **Utilities** on the **Definition** menu and then press Enter.

j. Enter **1** for Install definition and then press Enter. This action installs your changes to the WLM couple data set.

Note: If your installation already has service definitions defined on a couple data set, first merge your sample policy with the existing definitions. Otherwise, you can inadvertently lose all of your existing definitions. See your WLM administrator before installing the Web server sample policy into the couple data set.

k. Put the cursor under **Utilities** on the **Definition** menu and then press Enter.

l. Enter **3** for Activate service policy and then press Enter.

m. Exit the dialogs.

3. Ensure that WLM is in GOAL mode. Issue the following command in Spool Display and Search Facility (SDSF) to check if WLM is in GOAL mode:

```
/D WLM,SYSTEMS
```

You receive a response similar to the following. This response indicates that WLM is in GOAL mode:

```
RESPONSE=MVS076
 IWM025I  09.10.53  WLM DISPLAY 260
   ACTIVE WORKLOAD MANAGEMENT SERVICE POLICY NAME: WEB_RAL
   ACTIVATED: 2000/03/27  AT: 12:59:23  BY: USER1     FROM: MVS076
   DESCRIPTION: Web Service Policy for Raleigh
   RELATED SERVICE DEFINITION NAME: WEB_GOAL
   INSTALLED: 1999/09/08  AT: 10:30:38  BY: USER1     FROM: MVS085
   WLM VERSION LEVEL: LEVEL008
   *SYSNAME*  *MODE*  *POLICY*  *WORKLOAD MANAGEMENT STATUS*
   MVS076     GOAL    WEB_RAL   ACTIVE
```

If WLM is in COMPATABILITY mode instead of GOAL mode, issue the following command in SDSF to put WLM in GOAL mode:

```
/F WLM,MODE=GOAL
```

Step 2: Define the queue manager and queue server procedures.

1. Define the queue manager procedure. For this example modify the IMWEBSRV procedure described in "IMWHTTPD program" on page 400.

```
//IMWEBSRV PROC LEPARM='ENVAR("_CEE_ENVFILE=/etc/httpd.envvars")',
//   ICSPARM='-SN IMWEBSUB -r /etc/httpd.conf'
//*****************************************************************
//WEBSRV EXEC PGM=IMWHTTPD,REGION=0K,TIME=NOLIMIT,
//    PARM=('&LEPARM/&ICSPARM')
//*****************************************************************
//SYSIN    DD DUMMY
//OUTDSC   OUTPUT DEST=HOLD
//SYSTCPD  DD DSN=SYS1.TCPPARMS(TCPDATA@),DISP=SHR
//SYSPRINT DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//CEEDUMP  DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//
```

- **IMWEBSRV** is the queue manager procedure name. For consistency, name the procedure and the member that contains the procedure the same.
- **-SN** is the subsystem keyword.
- **IMWEBSUB** is the subsystem name. Make the subsystem name unique for each queue manager.

2. Define the procedure that is used by WLM to launch the queue servers. For this example the IMWIWM sample procedure described in "IMWIWM PROC (workload management)" on page 409, without alteration. The queue manager automatically passes the subsystem name and application environment name for each of the queue servers to the IMWIWM procedure. The queue manager procedure knows to call the IMWIWM procedure because the IMWIWM procedure name is specified on the 'Modify an Application Environment' panel in the WLM ISPF administrator dialogs. To see a sample of this panel with the IMWIWM procedure specified, go to "Configuring workload management to support the Web server" on page 228.

```
//IMWIWM  PROC IWMSN=,IWMAE=
//*********************************************************************
//*
//* ICSPARM ==> HTTP Server parameters
//* # WLM ApplEnv Queue Server
//* ICSPARM='-SN WEBSN1 -AE WEBHTML'
//*
//*  HTTP Server Parameters:
//*   -SN                        # WLM - subsystem name
//*   -AE                        # WLM - Application Envirnonment
//*
//*********************************************************************
//  SET SN='-SN '
//  SET AE=' -AE '
//  SET QQ='
//WEBSRV   EXEC PROC=IMWEBSRV,REGION=0K,TIME=NOLIMIT,
//   ICSPARM=&QQ&SN&IWMSN&AE&IWMAE&QQ
```

- **IMWEBSRV** is the queue manager procedure name. If you use a different procedure name for the queue manager, Change the PROC= value to match the queue manager procedure name.
- **ICSPARM** contains the parameters needed to start the queue server address space.
- **&IWMSN** is a symbolic parameter that automatically receives the subsystem name from the queue manager. This parameter is defined in your WLM policy for each application environment.

- **&IWMAE** is a symbolic parameter that automatically receives the application environment name from the queue manager. This parameter is defined in your WLM policy for each application environment.

3. Define the IMWEBSRV and IMWIWM procedures to the started task table, or the RACF started procedures table. In this example you define both procedures to the started task table:

```
RALTER STARTED IMWEBSRV.** STDATA(USER(WEBSRV))
RALTER STARTED IMWIWM.** STDATA(USER(WEBSRV))
SETROPTS RACLIST(STARTED) REFRESH
```

For more information, see "Step 5. Copy and customize the Web server PROC" on page 29.

Step 3: Set up the Web server configuration file to support a scalable server.

Set the following directives in your server configuration file, `httpd.conf`:

1. Define the application environment:

```
ApplEnv /*.html   WEBHTML    FAST
ApplEnv /*.gif    WEBCGI     FAST
ApplEnv /*        WEBDFLT    FAST
```

where:

- **/*.html** is the URI template for HTML requests.
- **WEBHTML** is the application environment for HTML requests. This name must match an application environment name in the WLM policy.
- **/*.gif** is the URI template for Common Gateway Interface (CGI) requests.
- **WEBCGI** is the application environment for CGI requests. This name must match an application environment name in the WLM policy.
- **/*** is the URI template for all requests that do not match one of the other ApplEnv directives
- **WEBDFLT** is the application environment for all requests that are not handled by either WEBHTML or WEBCGI. This name must match an application environment name in the WLM policy.
- **FAST** indicates the WLM transaction class.

Note: In this example you direct the requests to different queue servers based on whether these requests require HTML files or GIF files. However, you can route requests for more sophisticated reasons based on how you define the URI template on the ApplEnv directive. For instance, you can route requests for a billing application that come in on URI `/billing/*` to an application environment you call BILLC.

2. Define the logging and reporting directives:

```
AccessLog     /u/usera/logs/httpd-log
AgentLog      /u/usera/logs/agent-log
RefererLog    /u/usera/logs/referer-log
ErrorLog      /u/usera/logs/httpd-errors
CgiErrorLog   /u/usera/logs/cgi-error
AccessReportRoot /u/usera/reports
```

Step 4: Verify that your scalable server can successfully serve a request.

1. Start the Web server. To start the Web server from Spool Display and Search Facility (SDSF), type the following on the command line:

```
/S IMWEBSRV
```

2. Insure that the queue manager started successfully. Check the job log for the queue manager by selecting the job name of IMWEBSRV in SDSF. You receive an IMW3536I message if the queue manager started successfully. An example of this message follows.

   ```
   IMW3536I QM 50333520 0.0.0.0:80 IMWEBSUB * READY
   ```

   - **QM** represents the queue manager.
   - **50333520** is the process ID for the queue manager.
   - **0.0.0.0** indicates that the queue manager is not bound to a particular IP address.
   - **80** is the primary port on which the queue manager listens.
   - **IMWEBSUB** is the WLM subsystem name.
   - * indicates that no application environment name exists because this message is for a queue manager.
   - **READY** indicates that the queue manager has successfully initialized and is ready to serve requests.
3. Verify that your scalable server can successfully serve a request.
   a. Start a browser and enter the following URL. `Frntpage.html` is a default home page shipped with the server.

      ```
      http://your_server_name/Frntpage.html
      ```

      Once you make this request, the Web server returns an HTML file and a GIF file to your browser. WLM starts one queue server to handle HTML requests and another queue server to handle GIF requests.
   b. Check that the queue servers started successfully. In SDSF you have two queue servers, both with the job name of IMWIWM. Review the job logs for both. You receive an IMW3536I message in each of the job logs if each of the queue servers started successfully.

      For the queue server handling HTML requests, a message like the following appears:

      ```
      IMW3536I QS 67110992 0.0.0.0:80 IMWEBSUB WEBHTML READY
      ```

      where:
      - **QS** represents for the queue server.
      - **67110992** is the process ID for the queue server.
      - **0.0.0.0** indicates that the queue server is not bound to a particular IP address.
      - **80** is the primary port on which the queue server listens.
      - **IMWEBSUB** is the WLM subsystem name.
      - **WEBHTML** is the application environment name for the queue server.
      - **READY** indicates that the queue server has successfully initialized and is ready to serve requests.

      For the queue server handling CGI requests, a message like the following appears:

      ```
      IMW3536I QS 83888051 0.0.0.0:80 IMWEBSUB WEBCGI READY
      ```

      where:
      - **QS** represents the queue server.
      - **83888051** is the process ID for the queue server.
      - **0.0.0.0** indicates that the queue server is not bound to a particular IP address.

- **80** is the primary port on which the queue server is listening.
- **IMWEBSUB** is the WLM subsystem name.
- **WEBCGI** is the application environment name for the queue server.
- **READY** indicates that the queue server has successfully initialized and is ready to serve requests.

Note: The queue manager did not start the WEBDFLT queue server since all requests went to the WEBCGI queue server and WEBHTML queue server.

c. Check the logs to insure that the queue manager and queue servers successfully processed your request. In this example, the following logs were created under `/u/usera/logs`. For scalable server mode the Web server appends the subsystem name and the process ID to the log file name. The server also appends the application environment name if applicable. You can therefore determine which logs belong to the queue manager and which logs belong to a particular queue server.

   In this example one set of logs exists for the queue manager, one set of logs for the queue server handling HTML requests, and one set of logs for the queue server handling CGI requests.

```
agent-log.Oct242001.IMWEBSUB.50333520
agent-log.Oct242001.IMWEBSUB.WEBCGI.83888051
agent-log.Oct242001.IMWEBSUB.WEBHTML.67110992
cgi-error.Oct242001.IMWEBSUB.50333520
cgi-error.Oct242001.IMWEBSUB.WEBCGI.83888051
cgi-error.Oct242001.IMWEBSUB.WEBHTML.67110992
httpd-errors.Oct242001.IMWEBSUB.50333520
httpd-errors.Oct242001.IMWEBSUB.WEBCGI.83888051
httpd-errors.Oct242001.IMWEBSUB.WEBHTML.67110992
httpd-log.Oct242001.IMWEBSUB.50333520
httpd-log.Oct242001.IMWEBSUB.WEBCGI.83888051
httpd-log.Oct242001.IMWEBSUB.WEBHTML.67110992
referer-log.Oct242001.IMWEBSUB.50333520
referer-log.Oct242001.IMWEBSUB.WEBCGI.83888051
referer-log.Oct242001.IMWEBSUB.WEBHTML.67110992
```

   where:

   - **IMWEBSUB** is the subsystem name
   - **50333520** is the process ID for the queue manager.
   - **WEBCGI** is the application environment name associated with the queue server handling CGI requests.
   - **83888051** is the process ID associated with the queue server handling CGI requests.
   - **WEBHTML** is the application environment name associated with the queue server handling HTML requests.
   - **67110992** is the process ID associated with the queue server handling HTML requests.

   Note: If your scalable server receives enough requests, WLM starts additional instances of the WEBCGI queue server and the WEBHTML queue server. In this case, you see additional log files created for these queue servers. For example, if WLM started an additional WEBCGI queue server with process ID 88148275 and an additional WEBHTML queue server with process ID 44983266, the additional log files include the following:

```
agent-log.Oct242001.IMWEBSUB.WEBCGI.88148275
agent-log.Oct242001.IMWEBSUB.WEBHTML.44983266
cgi-error.Oct242001.IMWEBSUB.WEBCGI.88148275
cgi-error.Oct242001.IMWEBSUB.WEBHTML.44983266
httpd-errors.Oct242001.IMWEBSUB.WEBCGI.88148275
httpd-errors.Oct242001.IMWEBSUB.WEBHTML.44983266
httpd-log.Oct242001.IMWEBSUB.WEBCGI.88148275
httpd-log.Oct242001.IMWEBSUB.WEBHTML.44983266
referer-log.Oct242001.IMWEBSUB.WEBCGI.88148275
referer-log.Oct242001.IMWEBSUB.WEBHTML.44983266
```

For the WEBHTML application environment, the queue server httpd-log name closely resembles the following:

```
httpd-log.Oct242001.IMWEBSUB.WEBHTML.67110992
```

This log contains entries similar to the following. Each entry is split on two lines for printing purposes. Each entry appears on one line in your log file.

```
9.27.56.55 - - [24/Oct/2001:13:45:18 +0400]
   "GET /Frntpage.html HTTP/1.1" 401 278
9.27.56.55 - usera [24/Oct/2001:13:45:37 +0400]
   "GET /Frntpage.html HTTP/1.1" 200 0
```

The scalable server successfully processed the request because the queue server handling HTML requests returned the `Frntpage.html` file to the browser. This action is indicated by the 200 on the second entry.

For the WEBCGI application environment, the queue server httpd-log name closely resembles the following:

```
httpd-log.Oct242001.IMWEBSUB.WEBCGI.83888051
```

The log file contains an entry like the following. The entry is split on two lines for printing purposes. Each entry appears on one line in your log file.

```
0.0.0.0 - usera [24/Oct/2001:13:45:41 +0400]
   "GET /Admin/lgmast.gif HTTP/1.1" 200 0
```

The scalable server successfully processed the request because the queue server handling CGI requests returned the `lgmast.gif` file to the browser. This action is indicated by the 200 on the entry.

The httpd-log for the queue manager remains empty because the queue manager routed all requests to the queue servers.

Example 2: Setting up multiple scalable servers on your system

You can run either a single scalable server or simultaneously run multiple scalable servers on your system. You can, for instance, run multiple scalable servers if you are in a design phase and a test phase. In this case you can have multiple people that require the use of a scalable server environment tailored to their specific needs. This example shows how to implement multiple scalable servers.

In this example you set up a scalable server so that the queue manager routes HTML requests to one queue server and GIF requests to another queue server.

Before you begin:

- Ensure that you have Workload Management (WLM) installed and running on your system. For information on Workload Management, see *z/OS MVS*

Planning: Workload Management and *z/OS Programming: Workload Management Services*. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

- Ensure that the user IDs issuing RACF commands, issuing console commands, and accessing WLM ISPF administration dialogs have the proper authority.

Summary of tasks

In this example, you perform the following main tasks:

1. Modify the Web server sample Workload Management (WLM) policy and then load it into a WLM couple data set.
2. Define the queue manager and queue server procedures.
3. Set up the Web server configuration file to support a scalable server.
4. Verify that the scalable server can successfully serve a request.

Step 1: Modify the Web server's sample WLM policy and then load it into a WLM couple data set.

1. Permit the Web server to use WLM and the WLM administration policies by issuing the following RACF commands.

```
RDEFINE FACILITY MVSADMIN.WLM.POLICY UACC(NONE) NOTIFY(WEBSRV)
RDEFINE FACILITY BPX.WLMSERVER UACC(NONE) NOTIFY(WEBSRV)
PERMIT MVSADMIN.WLM.POLICY CLASS(FACILITY) ID(WEBSRV) ACCESS(UPDATE)
PERMIT BPX.WLMSERVER CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

2. Define a workload in the WLM ISPF administrator dialogs to manage your Web server work. This example starts with the sample WLM policy shipped with the Web Server to define the workload. The data set name containing the sample policy is called IMW.SIMWTBL1. The sample contains two application environments: WEBHTML and WEBCGI. You create a third application environment called WEBDFLT. You specify all three of these application environments on ApplEnv directives in the Web server configuration file. Define any application environments that you specify on ApplEnv directives under Application Environments in the WLM administrator dialogs. To view the sample policy in the WLM administrative dialogs, see "Configuring workload management to support the Web server" on page 228.

   To modify the Web server sample WLM policy and then activate it, complete the following steps:

   a. Start the WLM ISPF administrator dialogs by selecting **WLM** from the **ISPF Primary Option** menu.
   b. Press Enter to continue.
   c. Enter **1** to read a saved definition and then press Enter.
   d. Type in the data set name of the saved definition, **'IMW.SIMWTBL1'** and then press Enter.
   e. Enter **9** to select the Application Environment option on the **Definition** menu and then press Enter.
   f. Enter **1** under Action on the **Application Environment Selection** list, to create a new application environment name,and then press Enter.
   g. Complete the fields on the **Create an Application Environment** panel and then press **PF3** to save the results:
      - Enter the Application Environment: **WEBDFLT**.
      - Enter the Description: **handles remaining requests**.
      - Enter the Subsystem Type: **IWEB**.
      - Enter the Procedure Name: **IMWIWM**.

- Enter the Start Parameters: **IWMSN=&IWMSSNM,IWMAE=&WEBDFLT**.
- Keep the default of **1**, No Limit, for the limit on starting server address spaces for a subsystem instance.

h. Press **PF12** to return to the **Definition** menu.

i. Put the cursor under **Utilities** on the **Definition** menu and then press Enter.

j. Enter **1** for Install definition and then press Enter. This action installs your changes to the WLM couple data set.

Note: If your installation already has service definitions defined on a couple data set, first merge your sample policy with the existing definitions. Otherwise, you can inadvertently lose all of your existing definitions. See your WLM administrator before installing the Web server sample policy into the couple data set.

k. Put the cursor under **Utilities** on the **Definition** menu and then press Enter.

l. Enter **3** for Activate service policy and then press Enter.

m. Exit the dialogs.

3. Ensure that WLM is in GOAL mode. Issue the following command in Spool Display and Search Facility (SDSF) to check if WLM is in GOAL mode:

```
/D WLM,SYSTEMS
```

You receive a response similar to the following. This response indicates that WLM is in GOAL mode:

```
RESPONSE=MVS076
 IWM025I  09.10.53  WLM DISPLAY 260
   ACTIVE WORKLOAD MANAGEMENT SERVICE POLICY NAME: WEB_RAL
   ACTIVATED: 2000/03/27  AT: 12:59:23  BY: USER1     FROM: MVS076
   DESCRIPTION: Web Service Policy for Raleigh
   RELATED SERVICE DEFINITION NAME: WEB_GOAL
   INSTALLED: 1999/09/08  AT: 10:30:38  BY: USER1     FROM: MVS085
   WLM VERSION LEVEL: LEVEL008
   *SYSNAME*  *MODE*  *POLICY*  *WORKLOAD MANAGEMENT STATUS*
   MVS076     GOAL    WEB_RAL   ACTIVE
```

If WLM is in COMPATABILITY mode instead of GOAL mode, issue the following command in SDSF to put WLM in GOAL mode:

```
/F WLM,MODE=GOAL
```

Step 2: Define the queue manager and queue server procedures.

1. Define the queue manager procedure. For this example modify the IMWEBSRV procedure described in "IMWHTTPD program" on page 400. Make the member name, procedure name, the subsystem name, and the step name the same. Make them unique from one scalable server to the next. You might, for example, use a person's initials. Copy the IMWEBSRV procedure into a member called IMWEBxxx, where xxx represents the person's initials. Change the procedure name, subsystem name, and step name to IMWEBxxx. Insure that each user has a unique `httpd.envvars` file and `httpd.conf` file.

```
//IMWEBKDM PROC LEPARM='ENVAR("_CEE_ENVFILE=/u/kdm/httpd.envvars")',
//   ICSPARM='-SN IMWEBKDM -r /u/kdm/httpd.conf'
//*****************************************************************
//IMWEBKDM EXEC PGM=IMWHTTPD,REGION=0K,TIME=NOLIMIT,
//    PARM=('&LEPARM/&ICSPARM')
//*****************************************************************
//SYSIN    DD DUMMY
//OUTDSC   OUTPUT DEST=HOLD
//SYSTCPD  DD DSN=SYS1.TCPPARMS(TCPDATA@),DISP=SHR
```

```
//SYSPRINT DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//CEEDUMP  DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//
```

- **IMWEBKDM** is used for the queue manager procedure name, the subsystem name, and the job step. For consistency, name the member that contains the procedure the same name.
- **-SN** is the subsystem keyword.
- **_CEE_ENVFILE** is the environment variable on which you define the Web server environment variables file.
- **/u/userkdm/httpd.envvars** is the path and file name for the `httpd.envvars` file owned by a person with initials KDM.
- **-r** is the keyword on which you define the Web server configuration file.
- **/u/userkdm/httpd.conf** is the path and file name for the `httpd.conf` file owned by a person with initials KDM.

2. Define the procedure that is used by WLM to launch the queue servers. The queue manager automatically passes the subsystem name and application environment name for each of the queue servers to the IMWIWM procedure. The queue manager knows to call the IMWIWM procedure because the IMWIWM procedure name is specified on the 'Modify an Application Environment' panel in the WLM ISPF administrator dialogs. To see a sample of this panel with the IMWIWM procedure specified, go to "Configuring workload management to support the Web server" on page 228.

   For this example modify the IMWIWM procedure described in "IMWIWM PROC (workload management)" on page 409. The difference between the IMWIWM procedure used here and the one used for the single scalable server is the `PROC=` value. For the single scalable server you hard coded IMWEBSRV for the `PROC=` value. In this example, the MVS JCL interpreter substitutes the value for &IWMSN.

```
//IMWIWM  PROC IWMSN=,IWMAE=
//*********************************************************************
//*
//* ICSPARM ==> HTTP Server parameters
//* # WLM ApplEnv Queue Server
//* ICSPARM='-SN WEBSN1 -AE WEBHTML'
//*
//*  HTTP Server Parameters:
//*   -SN                        # WLM - subsystem name
//*   -AE                        # WLM - Application Envirnonment
//*
//*********************************************************************
//  SET SN='-SN '
//  SET AE=' -AE '
//  SET QQ='
//WEBSRV   EXEC PROC=&IWMSN,REGION=0K,TIME=NOLIMIT,
//   ICSPARM=&QQ&SN&IWMSN&AE&IWMAE&QQ
```

   where:

   - **ICSPARM** contains the parameters needed to start the queue server address space.
   - **&IWMSN** is a symbolic parameter that automatically receives the subsystem name from the queue manager. This parameter is used on PROC and ICSPARM. When it is used on the PROC parameter,**&IWMSN** represents the

procedure name for the queue manager. This parameter is defined in your WLM policy for each application environment.

- **&IWMAE** is a symbolic parameter that automatically receives the application environment name from the queue manager. This parameter is defined in your WLM policy for each application environment.

Note: Once you start your scalable server and make a request, you can display active users in SDSF. Queue servers all have the same job name of IMWIWM. However, you can distinguish the queue servers from one scalable server to the next because each has a different PROCSTEP name. The PROCSTEP name appears the same as the job name of the queue manager.

3. Define the IMWEBxxx and IMWIWM procedures to the started task table or the RACF started procedures table. In this example you define both procedures to the started task table. Since you run multiple scalable servers on your system, the queue manager procedure name varies from one scalable server to the next. However, the WEBSRV user ID and the IMWIWM procedure name remain the same. In this example the IMWEBKDM procedure owned by the person with initials KDM is defined to the started task table.

```
RALTER STARTED IMWEBKDM.** STDATA(USER(WEBSRV))
RALTER STARTED IMWIWM.** STDATA(USER(WEBSRV))
SETROPTS RACLIST(STARTED) REFRESH
```

For more information, see “Step 5. Copy and customize the Web server PROC” on page 29.

Step 3: Set up the Web server configuration file to support a scalable server.

Make sure the following are unique for each of the scalable servers running on your system:

- Logging and reporting files
- Port
- SSL Port, if you are running in secure mode

Set the following directives in your server configuration file, `httpd.conf`:

1. Define the application environment:

```
ApplEnv /*.html   WEBHTML    FAST
ApplEnv /*.gif    WEBCGI     FAST
ApplEnv /*        WEBDFLT    FAST
```

where:

- **/*.html** is the URI template for HTML requests.
- **WEBHTML** is the application environment for HTML requests. This name must match an application environment name in the WLM policy.
- **/*.gif** is the URI template for Common Gateway Interface (CGI) requests.
- **WEBCGI** is the application environment for CGI requests. This name must match an application environment name in the WLM policy.
- **/*** is the URI template for all requests that do not match one of the other ApplEnv directives
- **WEBDFLT** is the application environment for all requests that are not handled by either WEBHTML or WEBCGI. This name must match an application environment name in the WLM policy.
- **FAST** indicates the WLM transaction class.

Note: In this example you direct the requests to different queue servers based on whether these requests require HTML files or GIF files. However, you can route requests for more sophisticated reasons based on how you define the URI template on the ApplEnv directive. For instance, you can route requests for a billing application that come in on URI `/billing/*` to an application environment you call BILLC.

2. Define the logging and reporting directives. Make the path unique for each user:

```
AccessLog     /u/userkdm/logs/httpd-log
AgentLog      /u/userkdm/logs/agent-log
RefererLog    /u/userkdm/logs/referer-log
ErrorLog      /u/userkdm/logs/httpd-errors
CgiErrorLog   /u/userkdm/logs/cgi-error
AccessReportRoot /u/userkdm/reports
```

3. Make the port number unique for each user. In this case, the userkdm ID is using port 8220.

```
Port 8220
```

Note: This example runs in normal mode only, not secure mode. You do not need to alter the **sslport** directive.

Step 4: Verify that your scalable server can successfully serve a request.

1. Start the Web server. To start the Web server from Spool Display and Search Facility (SDSF), type the following on the command line:

```
/S IMWEBKDM
```

2. Insure that the queue manager started successfully. Check the job log for the queue manager by selecting the job name of IMWEBSRV in SDSF. You receive an IMW3536I message if the queue manager started successfully. An example of this message follows.

```
IMW3536I QM 50333520 0.0.0.0:8220 IMWEBKDM * READY
```

where:

- **QM** represents the queue manager.
- **50333520** is the process ID for the queue manager.
- **0.0.0.0** indicates that the queue manager is not bound to a particular IP address.
- **8220** is the primary port on which the queue manager listens.
- **IMWEBKDM** is the WLM subsystem name.
- * indicates that no application environment name exists because this message is for a queue manager.
- **READY** indicates that the queue manager has successfully initialized and is ready to serve requests.

3. Verify that your scalable server can successfully serve a request.
   a. Start a browser and enter the following URL. `Frntpage.html` is a default home page shipped with the server.

   `http://`*`your_server_name:port`*`/Frntpage.html`

   where **`port`** is 8220 for this example.

   Once you make this request, an HTML file and a GIF file will be returned to your browser. WLM will start one queue server to handle HTML requests and another queue server to handle GIF requests.

b. Check that the queue servers started successfully. In SDSF you have two queue servers, both with the job name of IMWIWM. Review the job logs for both. You receive an IMW3536I message in each of the job logs if each of the queue servers started successfully.

For the queue server handling HTML requests, a like the following appears:

```
IMW3536I QS 67110992 0.0.0.0:8220 IMWEBKDM WEBHTML READY
```

where:

- **QS** represents the queue server.
- **67110992** is the process ID for the queue server.
- **0.0.0.0** indicates that the queue server is not bound to a particular IP address.
- **8220** is the primary port on which the queue server listens.
- **IMWEBKDM** is the WLM subsystem name.
- **WEBHTML** is the application environment name for the queue server.
- **READY** indicates that the queue server has successfully initialized and is ready to serve requests.

For the queue server handling CGI requests, a message like the following appears:

```
IMW3536I QS 83888051 0.0.0.0:8220 IMWEBKDM WEBCGI READY
```

where:

- **QS** represents the queue server.
- **83888051** is the process ID for the queue server.
- **0.0.0.0** indicates that the queue server is not bound to a particular IP address.
- **8220** is the primary port on which the queue server listens.
- **IMWEBKDM** is the WLM subsystem name.
- **WEBCGI** is the application environment name for the queue server.
- **READY** indicates that the queue server has successfully initialized and is ready to serve requests.

Note: The queue manager did not start the WEBDFLT queue server since all requests went to the WEBCGI queue server and WEBHTML queue server.

c. Check the logs to insure that the queue manager and queue servers successfully processed your request. In this example, the following logs were created under `/u/userkdm/logs`. For scalable server mode the Web server appends the subsystem name and the process ID to the log's file name. If it is applicable, the server also appends the application environment name. You can therefore determine which logs belong to the queue manager and which logs belong to a particular queue server.

In this example one set of logs exists for the queue manager, one set of logs for the queue server handling HTML requests, and one set of logs for the queue server handling CGI requests.

```
agent-log.Oct242001.IMWEBKDM.50333520
agent-log.Oct242001.IMWEBKDM.WEBCGI.83888051
agent-log.Oct242001.IMWEBKDM.WEBHTML.67110992
cgi-error.Oct242001.IMWEBKDM.50333520
cgi-error.Oct242001.IMWEBKDM.WEBCGI.83888051
cgi-error.Oct242001.IMWEBKDM.WEBHTML.67110992
httpd-errors.Oct242001.IMWEBKDM.50333520
```

```
httpd-errors.Oct242001.IMWEBKDM.WEBCGI.83888051
httpd-errors.Oct242001.IMWEBKDM.WEBHTML.67110992
httpd-log.Oct242001.IMWEBKDM.50333520
httpd-log.Oct242001.IMWEBKDM.WEBCGI.83888051
httpd-log.Oct242001.IMWEBKDM.WEBHTML.67110992
referer-log.Oct242001.IMWEBKDM.50333520
referer-log.Oct242001.IMWEBKDM.WEBCGI.83888051
referer-log.Oct242001.IMWEBKDM.WEBHTML.67110992
```

where:

- **IMWEBKDM** is the subsystem name
- **50333520** is the process ID for the queue manager.
- **WEBCGI** is the application environment name associated with the queue server handling CGI requests.
- **83888051** is the process ID associated with the queue server handling CGI requests.
- **WEBHTML** is the application environment name associated with the queue server handling HTML requests.
- **67110992** is the process ID associated with the queue server handling HTML requests.

Note: If your scalable server receives enough requests, WLM starts additional instances of the WEBCGI queue server and the WEBHTML queue server. In this case, you see additional log files created for these queue servers. For example, if WLM started an additional WEBCGI queue server with process ID 88148275 and an additional WEBHTML queue server with process ID 44983266, the additional log files would include the following:

```
agent-log.Oct242001.IMWEBKDM.WEBCGI.88148275
agent-log.Oct242001.IMWEBKDM.WEBHTML.44983266
cgi-error.Oct242001.IMWEBKDM.WEBCGI.88148275
cgi-error.Oct242001.IMWEBKDM.WEBHTML.44983266
httpd-errors.Oct242001.IMWEBKDM.WEBCGI.88148275
httpd-errors.Oct242001.IMWEBKDM.WEBHTML.44983266
httpd-log.Oct242001.IMWEBKDM.WEBCGI.88148275
httpd-log.Oct242001.IMWEBKDM.WEBHTML.44983266
referer-log.Oct242001.IMWEBKDM.WEBCGI.88148275
referer-log.Oct242001.IMWEBKDM.WEBHTML.44983266
```

For the WEBHTML application environment, the queue server httpd-log name closely resembles the following:

```
httpd-log.Oct242001.IMWEBKDM.WEBHTML.67110992
```

This log contains entries similar to the following. Each entry is split on two lines for printing purposes. Each appears on one line in your log file.

```
9.27.56.55 - - [24/Oct/2001:13:45:18 +0400]
   "GET /Frntpage.html HTTP/1.1" 401 278
9.27.56.55 - userkdm [24/Oct/2001:13:45:37 +0400]
   "GET /Frntpage.html HTTP/1.1" 200 0
```

The scalable server successfully processed the request because the queue server handling HTML requests returned the `Frntpage.html` file to the browser. This action is indicated by the 200 on the second entry.

For the WEBCGI application environment, the queue server's httpd-log name closely resembles the following:

```
httpd-log.Oct242001.IMWEBKDM.WEBCGI.83888051
```

This log contains an entry similar to the following. The entry is split on two lines for printing purposes. This entry appears on one line in your log file.

```
0.0.0.0 - userkdm [24/Oct/2001:13:45:41 +0400]
   "GET /Admin/lgmast.gif HTTP/1.1" 200 0
```

The scalable server successfully processed the request because the queue server handling CGI requests returned the `lgmast.gif` file to the browser. This action is indicated by the 200 on the entry.

The httpd-log for the queue manager remains empty because the queue manager routed all requests to the queue servers.

Server Activity Monitor

Use the Server Activity Monitor to monitor Web server performance and status. The Server Activity Monitor allows you to view Web server activity statistics, network statistics, and access log entries. You can access and display this information remotely without being on the same machine that is running the Web server. This option provides significantly more information than opening the console window.

Enabling the Server Activity Monitor

By default, the Server Activity Monitor is enabled by the following Service directive in the Web server configuration file:

```
Service   /Usage* INTERNAL:UsageFn
```

Accessing statistics and usage information

To access Server Activity Monitor statistics from the Web server Configuration and Administration Forms:

1. Click **Configuration and Administration Forms** on the default Front page of the Web server.
2. In the **System Management** section, click **Server Activity Monitor**.

To update the statistics on a page, click **Refresh**.

To access usage reports directly, go to the following URLs:

- To monitor server activity:
  `http://your.server.name/Usage/Initial`
- To monitor the status of proxy requests:
  `http://your.server.name/Usage/proxylog`
- To monitor the network load:
  `http://your.server.name/Usage/Netstat`
- To show a dynamic view of the access log:
  `http://your.server.name/Usage/Logs`

Web server activity statistics

The following table shows the statistics included on the Activity Statistics page. Details follow the table.

Thread counts	Request statistics	Throughput statistics	Connection counts	Response Times (in seconds)
Threads running: 39	Requests processed: 80	Response time for local files: 1 second	Active inbound connections: 1	Service Plugins --Maximum: 0.057782 --Minimum: 0.012920 --Average: 0.022376
Threads idle: 38	Request errors: 5	Response time for proxied requests: <1 second	Active outbound connections: 0	CGI --Maximum: 20.838786 --Minimum: 4.970356 --Average: 14.001646
Maximum allowed threads: 39	Requests discarded: 3	Bytes received: 83K	Connections since last SMF: 7	DNS lookup --Maximum: 0.319905 --Minimum: 0.000004 --Average: 0.002186
Non-SSL Waiting Threads: 16	Requests proxied today: 9	Bytes sent: 790K		SSL Handshake --Maximum: 53.886848 --Minimum: 0.000065 --Average: 0.002186
SSL Waiting Threads: 16	Proxy cache hit rate: 0%	Unknown Bytes Received: 0 K		Proxy Response --Maximum: 0.821369 --Minimum: 0.209022 --Average: 0.421406
Async I/O Waiting Threads: 0	Responses processed: 85			
Msg Queue Waiting Threads: 0	Responses discarded: 0			

Thread counts

Threads running
Total number of threads available to do work. Threads running and Maximum allowed threads are usually the same number. However, if the Web server shuts down any threads, Threads running equals Maximum allowed threads minus the number of threads shut down since starting the Web server.

Threads idle
Number of threads not currently used

Maximum allowed threads
Maximum number of threads as specified in the Web server configuration file on the MaxActiveThreads directive; this value does not change unless the Web server is restarted with a new configuration file value for MaxActiveThreads.

Non-SSL Waiting Threads
Number of non-Secure Sockets Layer (SSL) threads available for use; if this value is 0, all non-SSL threads are allocated.

SSL Waiting Threads
Number of Secure Sockets Layer (SSL) threads available for use; if this value is 0, all SSL threads are allocated.

Async I/O Waiting Threads
If the Web server is running in Scalable Server mode, number of asynchronous I/O threads available for use; if this value is 0, all asynchronous I/O threads are allocated.

Msg Queue Waiting Threads
If the Web server is running in Scalable Server mode, number of message queue threads available for use; if this value is 0, all message queue threads are allocated.

Request statistics

Requests processed
Number of file requests that the Web server has successfully served; Requests processed + Request errors = Responses processed. For example, if a request is for the Web server home page and 4 GIF files, the number of requests processed will be 5.

Request errors
Number of file requests that the Web server responded to with an error; Request errors + Requests processed = Responses processed.

Requests discarded
Number of file requests sent to the Web server that are not valid

Requests proxied today
If configured as a proxy server, the number of file requests this server forwards to another server

Proxy cache hit rate
If configured as a proxy server, the percentage of proxy file requests cached

Responses processed
Total number of successful file responses sent; Responses processed = Requests processed + Request errors.

Responses discarded
Number of file responses the Web server was not able to send back to the client

Throughput statistics

Response time for local files
Average time to process a request for a file on the Web server

Response time for proxied requests
Average time to process a file that is forwarded to a proxy server

Bytes received
Total number of bytes of data sent to the Web server in requests

Bytes sent
Total number of bytes of data sent by the Web server in responses

Unknown bytes received
Total number of bytes of data that are not part of a request

Connection counts

Active inbound connections
Total number of connections from clients to the Web server; if the Web server is configured as a proxy server, the total number of connections from clients or other servers to this proxy server.

Active outbound connections
Total number of connections from the Web server to clients; if the Web server is configured as a proxy server, the total number of connections from this proxy server to clients or other servers.

Connections since last SMF
Total number of connections this Web server has provided since the last SMF record write; cumulative since the last SMF record was written

Response times (in seconds)

Service Plugins
Maximum, minimum, and average time it takes to complete customized application functions

CGI Maximum, minimum, and average time it takes to complete Common Gateway Interface (CGI) programs

DNS lookup
Maximum, minimum, and average time it takes to complete the search for a domain name in the Domain Name Server (DNS)

SSL Handshake
Maximum, minimum, and average time it takes to complete the exchange of security information (IDs, passwords, certificates) between the Web server and browser

Proxy Response
If configured as a proxy server, the maximum, minimum, and average time it takes to complete a transaction between a browser, this proxy server, and another server

Network activity statistics

Figure 1 shows an example of the network statistics that are displayed on the Network Status page.

Figure 1. Example of Network Status page statistics

Note: Incoming and outgoing data values include only data received and sent by the server.

Access log entries

The access log page displays the 20 most recent entries in the access log. For more information on the access log, go to "Tailoring the logs your server keeps" on page 189.

Simple Network Management Protocol

A network management system is an application that runs continuously and is used to monitor, reflect status of, and control a network. Simple Network Management Protocol (SNMP) is the network management standard. It communicates management information with devices in a network. The network devices typically have an SNMP ***agent*** and one or more subagents. The SNMP agent talks to the ***network management station*** or responds to command line SNMP requests. The SNMP ***subagent*** retrieves and updates data and gives that data to the SNMP agent to communicate back to the requester.

The HTTP Server provides an SNMP ***management information base*** (***MIB***) and SNMP subagent so you can use any SNMP-capable network management system, such as TME 10 NetView, TME 10 Distributed Monitoring, or HP OpenView, to monitor your server's health, throughput, and activity. The MIB data describes the Web server being managed, reflects current and recent server status, and provides server statistics.

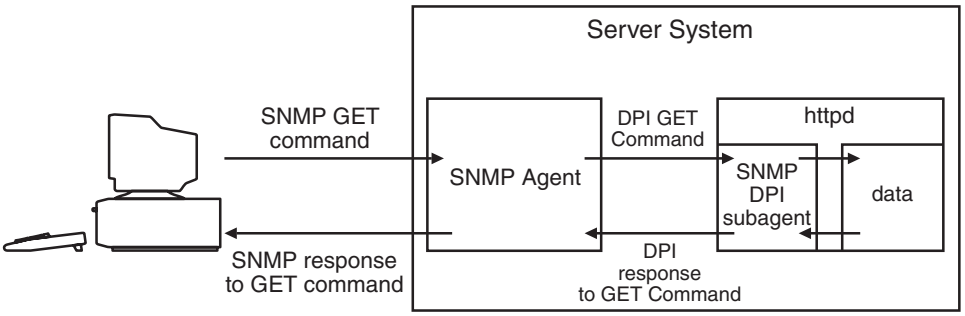
The network management system retrieves MIB values from other devices. It then can notify you if specified threshold values are exceeded. You can now proactively tune or fix server problems before they become server outages.

SNMP commands and protocol

Every device that is managed or that manages must have an SNMP agent. The user, management system, or programmer sends a GET command to the SNMP agent. In turn, this SNMP agent sends a GET command to retrieve the specified MIB variable values from a subagent responsible for those MIB variables.

The HTTP Server SNMP support includes an SNMP subagent that uses Distributed Protocol Interface (DPI) capability. DPI is an interface between an SNMP agent and its subagents.

The HTTP Server provides a subagent that updates and retrieves MIB data. The DPI subagent responds with the appropriate MIB data when the SNMP agent sends a GET command. The SNMP agent communicates the data to the network management station. The network management station can notify you if specified threshold values are exceeded.



Object IDs and variable names for the HTTP Server MIB

The HTTP Server MIB is modeled after the IETF WWW MIB RPC. MIB layout includes Variable Name, Object ID, Type, and Description.

The following Variable Names and Object IDs are provided for SNMP support with the HTTP Server:

EntityDescription

Description
Identifies a server in character string form. This read-only value is not customizable.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.1.1

Type OCTET_STRING

Default value
An appropriate value for your server installation and platform.

EntityObjectID

Description
Identifies a particular server in machine-readable form, providing a globally unique name among other applications and versions. This read-only value is not customizable.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.2.1

Type OCTET_STRING

Default value
1.3.6.1.4.1.2.6.120.10.1

EntityContact

Description
Indicates who to contact if a problem or question about this running server arises. It is a character string and frequently contains the e-mail address of the on-site system administrator responsible for server maintenance. The value for EntityContact may be customized with the WebMasterEmail directive in the httpd.conf file.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.3.1

Type OCTET_STRING

Default value
webmaster

EntityProtocol

Description
Identifies the exact protocol and its version that a particular server supports. For a Web server, the protocol is HTTP. This read-only identifier is in machine-readable form and is not customizable.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1

Type OCTET_STRING

Default value
1.3.6.1.4.1.2.12.1

EntityProtocolVersion

Description
This character string identifies the protocol this server supports and the protocol version. Similar to EntityProtocol. This read-only value is not customizable.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.5.1

Type OCTET_STRING

Default value
HTTP/1.1

EntityName

Description
This character string provides the name of the host this Web server runs on. The value for EntityName may be customized with the HostName directive in the httpd.conf file. It is read-only and set by system-specific code at server initialization time.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.6.1

Type OCTET_STRING

Default value
www.raleigh.ibm.com

EntityAddress

Description
This character string provides the IP address of the host this Web server runs on. It is read-only and set by system-specific code at server initialization time.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.7.1

Type IpAddress

Default value
0.0.0.0

EntityPort

Description
This character string provides the port number this Web server listens to. It is read-only and set by system-specific code at server initialization time.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1

Type INTEGER

Default value
0

EntityType

Description
This integer differentiates between several server roles.

Possible values are:
1 Simple or normal HTTP server
2 Proxy server
3 Caching server
4 Caching proxy

It is read-only and set by system-specific code at server initialization time. The information is taken from the httpd configuration file.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.9.1

Type INTEGER

Default value
1

CurrentThreads

Description
Indicates how many threads the server has currently. The total number of active threads is the sum of the MIB values, applInboundAssociations and applOutboundAssociations. This information is read-only.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.10.1

Type Gauge32

Default value
0

MaxThreads

Description
Indicates the maximum number of threads the server can have in the thread pool. This is read-only information.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.11.1

Type INTEGER

Default value
50

MinThreads

Description
Obsolete. Indicates the minimum number of threads the server can have. This is read only information.

Object ID
1.3.6.1.4.1.2.6.120.1.1.1.1.1.12.1

Type INTEGER

Default value
1

SummaryTable

Description
Indicates the number of requests in each category issued by this server. These are cumulative values, counted over the life of the server process.

All server address spaces under workload management update one counter, which is accessible by the HTTP Server Queue Manager.

Table object ID
1.3.6.1.4.1.2.6.120.1.1.2.1.1

Type Table

SummaryRequests

Description
Indicates the total number of requests the server received plus the total number of requests the server generated (for example, as a proxy server). This read-only information is updated as the server runs.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.1.1.1.1

Type COUNTER32

Default value
0

SummaryRequestErrors

Description
Indicates the total number of request errors detected by the server. This read-only information is updated as the server runs.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.1.1.2.1

Type COUNTER32

Default value
0

SummaryRequestDiscards

Description
Indicates the total number of requests discarded by the server (for any reason). This read-only information is updated as the server runs.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.1.1.3.1

Type COUNTER32

Default value
0

SummaryResponses

Description
Indicates the total number of responses generated or received by this server. This read-only information is updated as the server runs.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.1.1.4.1

Type COUNTER32

Default value
0

SummaryResponseDiscards

Description
Indicates the total number of responses discarded by the server. This read-only information is updated as the server runs.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.1.1.5.1

Type COUNTER32

Default value
0

SummaryInUnknowns

Description
Indicates the total number of unknown messages received by this server. This read-only information is updated as the server runs.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.1.1.6.1

Type COUNTER32

Default value
0

SummaryInBytes

Description
Indicates the total number of bytes received by this server. This read-only information is updated as the server runs.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.1.1.7.1

Type COUNTER32

Default value
0

SummaryOutBytes

Description
Indicates the total number of bytes sent out by this server. This read-only information is updated as the server runs.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.1.1.8.1

Type COUNTER32

Default value
0

ResponseSummaryTable

Description
Indicates the number of responses in each category issued by this server. These are cumulative values, counted over the life of the server process. These values are not affected by any changes to the configuration of the HTTP Server.

All server address spaces under workload management update one counter, which is accessible by the HTTP Server Queue Manager.

Table object ID
1.3.6.1.4.1.2.6.120.1.1.2.2.1.2

Type Table

ResponseError200Level

Description
Indicates the number of Error 200 responses (Positive Completion responses) issued by this server. This is a cumulative value, counted over the life of the Webserver process.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.2.1.2.1

Type Counter32

Default value
0

ResponseError300Level

Description
Indicates the number of Error 300 responses (Positive Intermediate Completion responses) issued by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.2.1.2.2

Type Counter32

Default value
0

ResponseError400Level

Description
Indicates the number of Error 400 responses (Transient Negative Completion responses) issued by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.2.1.2.3

Type Counter32

Default value
0

ResponseError500Level

Description
Indicates the number of Error 500 responses (Permanent Negative Completion responses) issued by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.2.1.2.4

Type Counter32

Default value
0

CacheTable

Description
Indicates information about the cache of this server. These values are affected by the cache settings in the configuration file.

All Webserver address spaces under workload management update one counter, which is accessible by the HTTP Server Queue Manager.

Table object ID
1.3.6.1.4.1.2.6.120.1.1.2.3.1

Type Table

CacheKBytesRead

Description
Indicates the number of kilobytes read from the cache of this server. This is a cumulative value, counted over the life of the server process.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.3.1.1.1

Type Counter32

Default value
0

CacheBytesRead

Description
Indicates the number of bytes read from the cache of this server. This is a cumulative value, counted over the life of the server process.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.3.1.2.1

Type Counter32

Default value
0

CacheHits

Description
Indicates the number of requests for files stored in the cache of this server. This is a cumulative value, counted over the life of the server process.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.3.1.3.1

Type Counter32

Default value
0

CacheRamInUse

Description
Indicates the number of bytes of RAM used by the cache of this server.

This is a snapshot value for the HTTP Server Queue Manager only.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.3.1.4.1

Type Counter32

Default value
0

CachedFiles

Description
Indicates the number of files in the cache of this server. This is a cumulative value, counted over the life of the server process.

This is a snapshot value for the HTTP Server Queue Manager only.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.3.1.5.1

Type Counter32

Default value
0

TotalTimeouts

Description
Indicates the total number of timeouts on the server. This read-only information is updated as the server runs. This is a cumulative value, counted over the life of the server process. This value is not affected by any changes to the configuration of the HTTP Server.

All server address spaces under workload management update one counter, which is accessible by the HTTP Server Queue Manager.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.4.0

Type Counter32

Default value
0

LastTimeoutEntityIndex

Description
This value is for future extensibility and provides support for the Application Table. This read-only value is always 1 and is not customizable.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.5.0

Type INTEGER

Default value
1

LastTimeoutRemoteAddress

Description
Provides the IP address of the machine that timed out last. This read-only value is updated by server code as the server runs. This value is not affected by any changes to the configuration of the HTTP Server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.6.0

Type IpAddress

Default value
0.0.0.0

RequestTable

Description
Indicates the number of requests for each method received by this server. These are cumulative values, counted over the life of the server process. These values are not affected by any changes to the configuration of the HTTP Server.

All Webserver address spaces under workload management update one counter, which is accessible by the HTTP Server Queue Manager.

Table object ID
1.3.6.1.4.1.2.6.120.1.1.2.7.1.2

Type Table

Default value
0

RequestGetMethod

Description
Indicates the number of GET requests received by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.7.1.2.1.1

Type Counter32

Default value
0

RequestHeadMethod

Description
Indicates the number of HEAD requests received by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.7.1.2.1.2

Type Counter32

Default value
0

RequestPostMethod

Description
Indicates the number of POST requests received by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.7.1.2.1.3

Type Counter32

Default value
0

RequestCGIMethod

Description
Indicates the number of CGI requests received by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.7.1.2.1.4

Type Counter32

Default value
0

RequestGWAPIMethod

Description
Indicates the number of the HTTP Server API (GWAPI) requests received by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.7.1.2.1.5

Type Counter32

Default value
0

ResponseTable

Description
Indicates the number of responses with these codes sent by this server. This is a cumulative value, counted over the life of the server process. This value is not affected by any changes to the configuration of the HTTP Server.

All Webserver address spaces under workload management update one counter, which is accessible by the HTTP Server Queue Manager.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.8.1.2

Type Table

Default value
0

ResponseError200

Description
Indicates the number of Error 200 responses (OK responses) sent by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.8.1.2.1.200

Type Counter32

Default value
0

ResponseError302

Description
Indicates the number of Error 302 responses (Moved Temporarily responses) sent by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.8.1.2.1.302

Type Counter32

Default value
0

ResponseError401

Description
Indicates the number of Error 401 responses (Unauthorized responses) sent by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.8.1.2.1.401

Type Counter32

Default value
0

ResponseError403

Description
Indicates the number of Error 403 responses (Forbidden responses) sent by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.8.1.2.1.403

Type Counter32

Default value
0

ResponseError404

Description
Indicates the number of Error 404 responses (Not Found responses) sent by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.8.1.2.1.404

Type Counter32

Default value
0

ResponseError407

Description
Indicates the number of Error 407 responses (Proxy Unauthorized) sent by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.8.1.2.1.407

Type Counter32

Default value
0

ResponseError500

Description
Indicates the number of Error 500 responses (Internal Server Error responses) sent by this server.

Object ID
1.3.6.1.4.1.2.6.120.1.1.2.8.1.2.1.500

Type Counter32

Default value
0

ApplicationData

Description
RFC 1565.

applName

Description
The name that the network service application is known by. This read-only value is not customizable.

Object ID
1.3.6.1.2.1.27.1.1.2.1

Type OCTET_STRING

Default value
HTTP Server *platform*

applDirectoryName

Description
The X.500 name for Web server. This read-only value is not customizable and is currently not supported by the HTTP Server.

Object ID
1.3.6.1.2.1.27.1.1.3.1

Type OCTET_STRING

Default value
Not available

applVersion

Description
The version of software the server is running. This character value is read-only and not customizable.

Object ID
1.3.6.1.2.1.27.1.1.4.1

Type OCTET_STRING

Default value
5.0

applUptime

Description
This value is how long the server has been up. This is a read-only value.

For workload management, this value is for the HTTP Server Queue Manager.

Object ID
1.3.6.1.2.1.27.1.1.5.1

Type TimeTicks

Default value
0

applOperStatus

Description
Indicates the operational status of the Web server. The HTTP Server sets this value to *up* at server startup. It is currently a read-only value.

For workload management, this value is for the HTTP Server Queue Manager.

Additional standardized values for this MIB variable include down, halted, congested, and restarting. These values may be used in the future.

Standardized values include:

1 Up - indicates that the server is operational and available.

2 Down - indicates that the Web server is not available.

3 Halted - indicates that the Web server is operational but not available.

4 Congested - indicates that the server is operational but no additional inbound associations can be accommodated.

5 Restarting - indicates that the server is currently unavailable but is in the process of restarting and will be available soon.

Object ID
1.3.6.1.2.1.27.1.1.6.1

Type INTEGER

Default value
1

applLastChange

Description
Indicates how long from when the server came up (applUptime) that the applOperStatus changed. Currently this will always be *0* because applOperStatus is only set to *up* at server startup. This is a read-only value.

For workload management, this value is for the HTTP Server Queue Manager.

Object ID
1.3.6.1.2.1.27.1.1.7.1

Type TimeTicks

Default value
0

applInboundAssociations

Description
Indicates the number of inbound connections currently running or how many threads are processing received requests. This is a read-only value. This value is not affected by any changes to the configuration of the HTTP Server.

For workload management, this value is for the HTTP Server Queue Manager.

Object ID
1.3.6.1.2.1.27.1.1.8.1

Type Gauge32

Default value
0

applOutboundAssociations

Description
Indicates the number of outbound connections that the server is currently handling or how many threads are processing outbound requests. This value is 0 if the server is not acting as a proxy server. This is a read-only value. This value is not affected by any changes to the configuration of the HTTP Server.

For workload management, this value is for the HTTP Server Queue Manager.

Object ID
1.3.6.1.2.1.27.1.1.9.1

Type Gauge32

Default value
0

applAccumulatedInboundAssociations

Description
Indicates the total number of server's inbound connections until this time. This is a read-only value. This value is cumulative and is not affected by any changes to the configuration of the HTTP Server.

For workload management, this value is for the HTTP Server Queue Manager.

Object ID
1.3.6.1.2.1.27.1.1.10.1

Type Gauge32

Default value
0

applAccumulatedOutboundAssociations

Description
Indicates the total number of server's outbound connections until this time. This value is 0 if the server is not acting as a proxy server. This is a read-only value. This value is cumulative and is not affected by any changes to the configuration of the Webserver.

For workload management, this value is for the HTTP Server Queue Manager.

Object ID
1.3.6.1.2.1.27.1.1.11.1

Type Counter32

Default value
0

applLastInboundActivity

Description
Indicates the time since applUptime that the last inbound connection was made. This is a read-only value. This value is not affected by any changes to the configuration of the HTTP Server.

For workload management, this value is for the HTTP Server Queue Manager.

Object ID
1.3.6.1.2.1.27.1.1.12.1

Type TimeTicks

Default value
0

applLastOutboundActivity

Description
Indicates the time since applUptime that the last outbound connection was made. This is a read-only value. This value is not affected by any changes to the configuration of the HTTP Server.

For workload management, this value is for the HTTP Server Queue Manager.

Object ID
1.3.6.1.2.1.27.1.1.13.1

Type TimeTicks

Default value
0

applRejectedInboundAssociations

Description
Indicates the total number of requests the server has rejected. This is a read-only value.This value is not affected by any changes to the configuration of the HTTP Server.

For workload management, this value is for the HTTP Server Queue Manager.

Object ID
1.3.6.1.2.1.27.1.1.14.1

Type Counter32

Default value
0

applFailedOutboundAssociations

Description
Indicates the total number of the server's outbound requests that failed. This is a read-only value.This value is not affected by any changes to the configuration of the Webserver.

For workload management, this value is for the HTTP Server Queue Manager.

Object ID
1.3.6.1.2.1.27.1.1.15.1

Type Counter32

Default value
0

Note: The timestamp values for the HTTP Server MIB variables, applLastChange, applLastInboundActivity, and applLastOutboundActivity, vary from RFC 1565. In RFC 1565, timestamps are relative to sysUpTime. These three timestamp values are relative to applUptime.

Defining a management information base file so that you can do queries with variable names

When you query the Simple Network Management Protocol (SNMP) agent, you can issue a command with a variable name or an object ID. To issue a command with a variable name such as EntityProtocol, you specify the variable along with its object ID and its type in the management information base (MIB) file, `/etc/mibs.data`. The file ships empty so that you can specify variable names. Following is an example of the `/etc/mibs.data` file:

```
# short name                        OID                 type
#-------------------------------------------------------------
EntityProtocol    1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1 octetstring
EntityProtVersion 1.3.6.1.4.1.2.6.120.1.1.1.1.1.5.1 octetstring
mysrvName         1.3.6.1.4.1.2.6.120.1.1.1.1.1.6.1 octetstring
myipaddress       1.3.6.1.4.1.2.6.120.1.1.1.1.1.7.1 ipaddress
mysrvport         1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1 integer
```

The first part of each object ID consists of a unique subagent number, which for the Web server is 1.3.6.1.4.1.2.6.120. The corresponding unique subagent name is ibmInternetConnSvr.

You can specify variable names that are documented in "Object IDs and variable names for the HTTP Server MIB" on page 254 or other names that you define. The `mysrvName, myipaddress, mysrvport` variable names are user defined.

Simple Network Management Protocol examples: running with a single Web server and a single Simple Network Management Protocol agent

These examples use the z/OS UNIX System Services **osnmp** command to query the Simple Network Management Protocol (SNMP) agent for network management information. For more information on the **osnmp** command and other methods of querying the SNMP agent, see *z/OS Communications Server IP System Administrator's Commands* for V1R2 and releases above V1R2 or *z/OS Communications Server IP User's Guide* for V1R1. To access these books on the Web, go to the z/OS Book Server Web site at URL:
`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

The **osnmp** commands in these examples use the `findname` parameter and the `get` parameter. The `findname` parameter requires the object ID as an operand. The `findname` query returns a variable from the `/etc/mibs.data` file. If the file is empty, the SNMP agent returns the registered name of the subagent concatenated in front of the remaining object ID. The `get` parameter can use either a variable name from the `/etc/mibs.data` file, or an object ID.

Example 1: Using an empty /etc/mibs.data file

This example shows how to obtain the protocol and the protocol version that the Web server supports. For more information on the protocol, see `EntityProtocol` in "Object IDs and variable names for the HTTP Server MIB" on page 254. Use an empty `/etc/mibs.data` file.

The `findname` query returns the name of the registered subagent, ibmInternetConnSvr, concatenated with the remainder of the object ID.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp findname 1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1
```

The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1 found as: ibmInternet
    ConnSvr.1.1.1.1.1.4.1
```

Note: The response appears on two lines for printing purposes.

The following `get` query uses the result from the `findname` query to request the protocol information.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get ibmInternetConnSvr.1.1.1.1.1.4.1
```

The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1 =
  1.3.6.1.4.1.2.12.1
```

Note: The response appears on two lines for printing purposes.

The following `get` query uses the object ID associated with the `EntityProtocol` variable to determine the protocol information. You can do this query without first doing a `findname` query since it uses the object ID.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get 1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1
```

The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1 =
  1.3.6.1.4.1.2.12.1
```

Note: The response appears on two lines for printing purposes.

Example 2: Using an updated /etc/mibs.data file

This example shows how to obtain the protocol and the protocol version that the Web server supports. For more information on the protocol, see `EntityProtocol` in "Object IDs and variable names for the HTTP Server MIB" on page 254. Use the `/etc/mibs.data` file in "Defining a management information base file so that you can do queries with variable names" on page 269.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp findname 1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1
```

The SNMP agent returns the following:

```
>1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1 found as:
  EntityProtocol.1
```

Notes:

1. The `findname` query returns *variable name*.1.
2. The response appears on two lines for printing purposes.

The following `get` query uses `EntityProtocol.1` returned in the `findname` query to determine the protocol information.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get EntityProtocol.1
```

The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1.1 =
  1.3.6.1.4.1.2.12.1
```

Note: The response appears on two lines for printing purposes.

The following `get` query uses the object ID associated with the `EntityProtocol` variable to determine the protocol information. You can do this query without first doing a `findname` query since it uses the object ID. This query does not use an `/etc/mibs.data` file, but one can exist.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get  1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1
```

The SNMP agent returns the following:

```
1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1 =
  1.3.6.1.4.1.2.12.1
```

Note: The response appears on two lines for printing purposes.

The following `get` query uses the `EntityProtocol` variable name found in the `/etc/mibs.data` file. You can do this query without first doing a `findname` query.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get EntityProtocol
```

The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.4.1 =
  1.3.6.1.4.1.2.12.1
```

Note: The response appears on two lines for printing purposes.

Example 3: Employing user defined variables in the /etc/mibs.data file

This example shows how to obtain the Web server port. Employ the user variables in the `/etc/mibs.data` file in "Defining a management information base file so that you can do queries with variable names" on page 269.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
osnmp findname 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1
```

The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1 found as:
  mysrvport.1
```

Notes:

1. The `findname` query returns *variable name*.1.
2. The response appears on two lines for printing purposes.

The following `get` query uses `mysrvport.1` returned in the `findname` query to determine the protocol information.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
osnmp get mysrvport.1
```

The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1.1
  = 8181
```

where

- **8181** is the server port.

Note: The response appears on two lines for printing purposes.

The following `get` query uses the `mysrvport` variable in the `/etc/mibs.data` file to determine the server port. You can do this query without first doing a `findname` query.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get mysrvport
```

The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1.1 =
  8181
```

where

- **8181** is the server port.

Note: The response appears on two lines for printing purposes.

The following `get` query uses the object ID associated with the `mysrvport` variable to determine the server port. You can do this query without first doing a `findname` query since it uses the object ID. This query does not use an `/etc/mibs.data` file, but one can exist.

Type the `osnmp` command on the z/OS UNIX System Services command line:

```
> osnmp get 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1
```

The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1 = 8181
```

where

- **8181** is the server port.

Note: The response appears on two lines for printing purposes.

The following `get` query uses the object ID associated with the `mysrvport` variable to determine the protocol information. The `mysrvport` object ID is in the `/etc/mibs.data` file. You can do this query without first doing a `findname` query since it uses the object ID.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get
  1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1
```

The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1
  = 8181
```

where

- **8181** is the server port.

Note: The query and the response each appear on two lines for printing purposes.

Simple Network Management Protocol examples: running multiple Web servers, each with a unique Simple Network Management Protocol agent

If you run multiple Web servers and enable Simple Network Management Protocol (SNMP) for each Web server subagent, you have a separate SNMP agent running for each Web server. The SNMP port for receiving SNMP requests is unique for each agent and subagent combination.

Set up your environment as follows. The steps below are optional for the default port, required for user defined ports.

- Set the SNMP port for the agent by adding the `–p` argument to the parameter line of the SNMP procedure. The following example specifies the port as 171:

  ```
  //      PARM='POSIX(ON) TRAP(OFF) ALL31(ON)/ -p 171 '
  ```

- Set the SNMP port for the subagent on the `SNMP_PORT` environment variable. You can set this environment variable in the Web server `httpd.envvars` file. The following example sets the port to 171 in the `httpd.envvars` file:

  ```
  SNMP_PORT=171
  ```

- Include the `–h` parameter on the **osnmp** command when you issue queries. On this parameter you specify the target host to which you want to send requests. You can represent the host by a host name with its unique SNMP port, an Internet Protocol (IP) address with its unique SNMP port, or a symbolic name. For more information on the `–h` parameter, see *z/OS Communications Server IP System Administrator's Commands* for V1R2 and releases above V1R2 or *z/OS Communications Server IP User's Guide* for V1R1. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.
- If you use a symbolic name on the `–h` parameter, update the **winSNMPname** field and the **targetAgent** field in the `/etc/snmpv2.conf` file. For information on these fields, see *z/OS Communications Server IP Configuration Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

These examples assume that you have two Web servers running.

Use the following `/etc/snmpv2.conf` file for the examples that use the **winSNMPname** field on the `–h` parameter:

```
HOST161 9.37.65.142:161 snmpv1
HOST171 9.37.65.142:171 snmpv1
```

where:

- **HOST161** is the **winSNMPname** field.
- **HOST171** is the **winSNMPname** field.
- **9.37.65.142** is the IP address.
- **161** is the unique SNMP port.
- **171** is the unique SNMP port.
- **9.37.65.142:161** is the **targetAgent** field.
- **9.37.65.142:171** is the **targetAgent** field.
- **snmpv1** is the administrative model supported by the **targetAgent** field.

These examples use the z/OS UNIX System Services **osnmp** command to query the Simple Network Management Protocol (SNMP) agent for network management information. For more information on the **osnmp** command and other methods of querying the SNMP agent, see *z/OS Communications Server IP System Administrator's Commands* for V1R2 and releases above V1R2 or *z/OS Communications Server IP User's Guide* for V1R1. To access these books on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

The `osnmp` commands in these examples use the `get` parameter. The `get` parameter can use either a variable name from the `/etc/mibs.data` file or an object ID.

Server 1 examples

Assume the first server has the following characteristics:

- The Web server IP address is 9.37.65.142.
- The Web server port is 8181.
- The SNMP port is the default 161 port.

- The –p parameter is not coded in the SNMP procedure.
- The SNMP_PORT environment variable is not set in the httpd.envvars file.

The following get query uses the –h parameter with the **winSNMPname** symbolic name of HOST161.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get -h HOST161
 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1
```

Note: The **osnmp** command appears on two lines for printing purposes.
The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1 = 8181
```

The following get query uses the –h parameter with the Web server IP address and Web server port.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get -h 9.37.65.142:161
  1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1
```

Note: The **osnmp** command appears on two lines for printing purposes.
The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1 = 8181
```

Since the following get query does not use the –h parameter, it uses the default 161 port.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get
  1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1
```

Note: The **osnmp** command appears on two lines for printing purposes.
The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1 = 8181
```

Server 2 examples

Assume the second server has the following characteristics:

- The Web server IP address is 9.37.65.142.
- The Web server port is 8191.
- The SNMP port is the user defined 171 port.
- The –p parameter is coded in the SNMP procedure with 171.
- The SNMP_PORT environment variable is set in the httpd.envvars file.

The following get query uses the –h parameter with the symbolic name of **HOST171**.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get -h HOST171
  1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1
```

Note: The **osnmp** command appears on two lines for printing purposes.
The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1 = 8191
```

The following `get` query uses the `–h` parameter with the Web server IP address and Web server port.

Type the **osnmp** command on the z/OS UNIX System Services command line:

```
> osnmp get -h 9.37.65.142:171
  1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1
```

Note: The **osnmp** command appears on two lines for printing purposes.
The SNMP agent returns the following:

```
> 1.3.6.1.4.1.2.6.120.1.1.1.1.1.8.1 = 8191
```

Creating an e-mail address to receive Simple Network Management Protocol problem reports

The HTTP Server provides a default e-mail address, `webmaster`, for use by management applications that send e-mail as a result of threshhold exceptions. The HTTP Server does not send such e-mail notices. Use the WebMasterEmail directive to customize the mail address. The typical format for this value is `user@rootname`. For more information about the WebMasterEmail directive, see "WebMasterEmail - Create an e-mail address to receive SNMP problem reports" on page 564.

Providing a security password for Simple Network Management Protocol

You can create community names (passwords). The SNMP community name authorizes a user to view the performance variables monitored by SNMP for a particular community of servers. The system administrator defines which variables from which servers can be viewed when a password is entered. If you change the SNMP community name, be sure to also change the community name specified in the file named `/etc/snmpd.conf`.

The default SNMP community name is `public`. This is kept in a dataset named *`hlq`*`.PW.SRC` where *hlq* is the high level qualifier fo TCP/IP datasets. This dataset is accessible from z/OS UNIX System Services. The default entry is:

```
public 0.0.0.0 0.0.0.0
```

The search order is:

- /etc/pw.src
- The dataset specified on the SYSPWSRC DD statement in the agent procedure
- *jobname*.PW.SRC, where *jobname* is the name of the job used to start the SNMP agent
- STS1.TCPPARMS(PWSRC)
- *hlq*.PW.SRC, where *hlq* either defaults to TCPIP or is specified on the DATASETPREFIX statement in the TCPIP.DATA file.

The default entry can be used when testing the default community name, `public`.

The *hlq* value can be determined by:

1. Looking at the JCL for the started procedure for the SNMP agent for MVS. This is usually 'SYS1.PROCLIB(SNMPD)'.
2. Looking in the dataset referred to by the SYSTCPD DD statement. This is usually 'SYS1.TCPPARMS(TCPDATA)'. In this dataset member is a keyword

DATASETPREFIX. The value for DATASETPREFIX is what *hlq* equals. For example, if the statement is `DATASETPREFIX TCP`, then the community names are in TCP.PW.SRC.

Use the SNMPCommunityName directive to define the community name used between the HTTP Server DPI subagent and the SNMP agent. For more information about the SNMPCommunityName directive, see "SNMPCommunityName - Provide a security password for SNMP" on page 564.

Enabling and disabling SNMP support

Use the SNMP directive to enable or disable SNMP support. To enable SNMP support, change the SNMP value to on. The default SNMP value is off.

For more information about the SNMP directive, see "Turning the SNMP support on and off from the IMWHTTPD program".

Turning the SNMP support on and off from the IMWHTTPD program

Use these flags to turn the SNMP support in the HTTP Server on and off.

The `-snmp` flag turns the SNMP support on. The `-nosnmp` flag turns the SNMP support off.

This overrides what is defined in the httpd.conf file.

For more information about the IMWHTTPD program, see "IMWHTTPD program" on page 400.

System Management Facilities

With the System Management Facilities (SMF), you can request that configuration and performance data be recorded to SMF datasets. With this recorded configuration and statistical data, you can monitor Web server health, throughput, and activity.

If you want other data to be recorded in an SMF record, you can use the sample SMF plug-in that is shipped with the server. This sample is located in */install_path/samples/API/smflog.c*.

Configuration record data is taken from the server configuration file, http.conf, and is written after the server daemon is fully initialized. Performance record data is accumulated continuously and written at intervals defined in the http.conf file by the SMFRecordingInterval directive.

Note: When the logging queue is full, the logging thread checks to see if it needs to write any SMF records, based on whether the SMF recording interval has expired. If activity on the server is minimal, the logging queue fills slowly. In this case, the SMF records are written less frequently. You may want to adjust the interval specified by the SMFRecordingInterval directive.

When running under workload management (WLM), the HTTP Server Queue Manager writes sum totals for the Queue Manager and Queue Servers to SMF.

To write SMF records, the SMF application must be enabled to accept 103 type records. If the SMF application is not enabled, the server turns off SMF recording. To use Console Modify to turn SMF recording back on, see "Turning SMF on and off with the z/OS operator console MODIFY command". To enable SMF to receive the records, refer to *z/OS MVS System Management Facilities*.

Your systems programmer must also permit to the user ID running httpd READ permission for the RACF BPX.SMF facility class.

For information on how to interpret SMF record headers and layouts, using and configuring SMF, dumping the SMF datasets, summarizing the information in the datasets, and getting record details, refer to the *z/OS MVS System Management Facilities* book.

Turning SMF support on and off from the IMWHTTPD program

Use these flags to turn the SMF support in the HTTP Server on and off.

The `-smf` flag turns the SMF support on. The `-nosmf` flag turns the SMF support off.

This overrides what is defined in the httpd.conf file.

For more information about the IMWHTTPD program, see "IMWHTTPD program" on page 400.

Turning SMF on and off with the z/OS operator console MODIFY command

You can use the z/OS operator console MODIFY command to turn SMF on or off.

Configuration and performance data is written to SMF records at the moment in which you turn SMF on with a command. After that point, it is written at time intervals specified by the SMFRecordingInterval directive. To turn SMF on with the operator console MODIFY command, enter:

```
F IMWEBSRV,APPL=-smf
```

To turn SMF off with the operator console MODIFY command, enter:

```
F IMWEBSRV,APPL=-nosmf
```

For more information about the z/OS operator console MODIFY command, see "z/OS MODIFY console command" on page 410.

Controlling the logging of information by SMF

You can choose to have either configuration record data or performance record data, both configuration and performance record data, or no data written to SMF datasets. You can also define how often SMF writes the continuously accumulated statistical information to SMF datasets.

To select the type of information to be written to the SMF dataset, you can use the Global Log File Configuration Settings form in the Configuration and Administration forms or the SMF directive. The default setting for the SMF directive is `none`.

To specify how frequently performance record information is written to SMF datasets, use the Global Log File Configuration Settings form in the Configuration and Administration forms or the SMFRecordingInterval directive. The default SMFRecordingInterval is `00:15`.

With the following exceptions the SMF performance record will be written on the interval specified by the SMFRecordingInterval:

- For the initial write on startup, the interval will be zero.
- For the termination write either on shutdown or restart, the interval will reflect the time since the last write.
- For the initial write on restart the interval will reflect the time since the restart termination write.

For more information about the Configuration and Administration Forms see "Using the Configuration and Administration forms" on page 50. For more information about the SMF directive, see "SMF - Specify the type of information that SMF records" on page 516. For more information about the SMFRecordingInterval directive, see "SMFRecordingInterval - Specify how often to record performance record information" on page 516.

SMF record formats

The SMF record formats are configuration and performance. Each SMF record format subtype has a different data content. The information in the SMF header is the same for both record formats.

- Configuration - Type 103 Subtype 01
  - SMF header
  - SMF data area
- Performance - Type 103 Subtype 02
  - SMF header
  - SMF data area

Self-defining section of the SMF record header

The SMF record header information is valid for both performance record data and configuration data.

Decimal Offsets	Hex Offsets	Name	Length	Format	Description
0	0	SMF103LEN	2	binary	Record Length
2	2	SMF103SEG	2	binary	Segment descriptor
4	4	SMF103FLG	1	binary	System Indicator
5	5	SMF103RTY	1	binary	Record Type 103 (X'67')
6	6	SMF103TME	4	binary	Time stamp
10	0A	SMF103DTE	4	packed	Date stamp
14	E	SMF103SID	4	EBCDIC	System Identifier
18	12	SMF103SSI	4	EBCDIC	Subsystem identifier
22	16	SMF103STY	2	binary	Record subtype (X'01' or X'02)

SMF configuration record data area (record type 103, subtype 01)

The httpd.conf configuration file provides all the information listed under the description. Use the field descriptions in that file for more detail.

Decimal Offsets	Name	Length	Format	Description
24	EntityNameLen	4	binary	Length of the Web server's host name
28	EntityName	var	EBCDIC	Web server's host name as specified in httpd.conf or Web server's default host name from TCP/IP
To determine the value for ENE, add 28 and the value for EntityNameLen, for example, ENE = 28+EntityNameLen.				
ENE	EntityAddressLen	4	binary	length of Web server's IP Address
ENE+4	EntityAddress	4	binary	Web server's IP address specified in httpd.conf or Web server's default IP address from TCP/IP
ENE+8	EntityPort	4	binary	port number on which the Web server is listening. The port number is the value specified on the Port directive.
ENE+12	serverType	4	binary	always 1
ENE+16	applVersionLen	4	binary	length of next field
ENE+20	applVersion	var	EBCDIC	version of server
To determine the value for AVE, add 20 and the value for applVersionLen, for example, AVE = ENE+20+applVersionLen.				
AVE	serverRootLen	4	binary	Length of server_root
AVE+	serverRoot	var	EBCDIC	directory for server_root
To determine the value for SRE, add 4 and the value for serverRootLen, for example, SRE = AVE+4+serverRootLen.				
SRE	doDNSLookUp	4	binary	dns lookup flag
SRE+4	maxContentBuf	4	binary	max size of content buffer
SRE+8	ThreadsMin	4	binary	minimum number threads
SRE+12	ThreadsMax	4	binary	maximum number of threads
SRE+16	IdleThreadTO	4	binary	timout for idle threads
SRE+20	ACLSettings	4	binary	ACL settings
SRE+24	UseMetaFiles	4	binary	meta file flag
SRE+28	DirAccess	4	binary	directory access flag
SRE+32	inputTO	4	binary	input timeout
SRE+36	outputTO	4	binary	output timeout
SRE+40	scriptTO	4	binary	script timeout
SRE+44	useGMT	4	binary	GMT flag
SRE+48	serverImbedsHtml	4	binary	server Imbeds HTML flag
SRE+52	secureType	4	binary	security type
SRE+56	sslPort	4	binary	security (SSL) port
SRE+60	normalMode	4	binary	normal mode flag
SRE+64	cacheOff	4	binary	cache flag
SRE+68	cache_max_k	4	binary	max k cache
SRE+72	cache_max_f	4	binary	max file to cache
SRE+76	cache_limit_1	4	binary	cache limit 1
SRE+80	cache_limit_2	4	binary	cache limit 2
SRE+84	cacheTimeMarginLen	4	binary	cache time margin field length

Decimal Offsets	Name	Length	Format	Description
SRE+88	cacheTimeMargin	4	binary	(long) cache time margin
SRE+92	cacheLockTOLen	4	binary	cache lock timeout field length
SRE+96	cacheLockTO	4	binary	(long) cache lock timeout
SRE+100	keepExpired	4	binary	keep expired flag
SRE+104	cacheNoConnect	4	binary	cache connect flag
SRE+108	gcDisabled	4	binary	garbage collection flag
SRE+112	gcDailyGCLen	4	binary	garbage collection interval
SRE+116	gcDailyGC	4	binary	(long) garbage collection interval
SRE+120	gcMemUsage	4	binary	garbage collection mem use
SRE+124	ProxySomething	4	binary	proxy flag

SMF performance record data area (record type 103, subtype 02)

The SNMP MIB provides all these performance numbers. For more details, use the descriptions in "Object IDs and variable names for the HTTP Server MIB" on page 254.

Decimal Offsets	Name	Length	Format	Description
24	EntityNameLen	4	binary	length of Web server's Address
28	EntityName	var	EBCDIC	Web server's host name as specified in httpd.conf or Web server's default host name from TCP/IP
To determine the value for ENE, add 28 and the value for EntityNameLen, for example, ENE = 28+EntityNameLen.				
ENE	EntityAddressLen	4	binary	length of Web server's IP Address
ENE+4	EntityAddress	4	binary	Web server's IP address specified in httpd.conf or Web server's default IP address from TCP/IP
ENE+8	EntityPort	4	binary	port number on which the Web server is listening. The port number is the value specified on the Port directive.
ENE+12	serverType	4	binary	always 1
ENE+16	applVersionLen	4	binary	length of next field
ENE+20	applVersion	var	EBCDIC	version of server
To determine the value for AVE, add 20 and the value for applVersionLen, for example, AVE = ENE+20+applVersionLen.				
AVE	TotalCurrentThreads	4	binary	number of threads currently used; represents the number of threads currently in use.
AVE+4	MaxThread	4	binary	maximum number of threads defined; static number
AVE+8	Request	4	binary	number of requests received; cumulative from Web server startup
AVE+12	RequestErrors	4	binary	number of request errors received; cumulative from Web server startup
AVE+16	RequestDiscards	4	binary	number of requests discarded; cumulative from Web server startup

Decimal Offsets	Name	Length	Format	Description
AVE+20	Responses	4	binary	number of responses sent; cumulative from Web server startup
AVE+24	ResponseDiscard	4	binary	number of responses discarded; cumulative from Web server startup
AVE+28	InBytes	4	binary	number of bytes received; cumulative from Web server startup
AVE+32	OutBytes	4	binary	number of bytes sent; cumulative from Web server startup
AVE+36	InUnknowns	4	binary	number of unknown type bytes received; cumulative from Web server startup
AVE+40	TotalTimeOuts	4	binary	number of timeouts since startup; cumulative from Web server startup
AVE+44	KBytesReadFromCache	4	binary	number of kilobytes read from the proxy cache; cumulative from Web server startup; KbytesReadFromCache x 1024 + BytesReadFromCache = total bytes read from the proxy cache
AVE+48	BytesReadFromCache	4	binary	number of bytes read from the proxy cache; cumulative from Web server startup; KbytesReadFromCache x 1024 + BytesReadFromCache = total bytes read from the proxy cache
AVE+52	CacheHits	4	binary	number of proxy cache hits; cumulative from Web server startup
AVE+56	BytesCacheRamInUse	4	binary	number of bytes of proxy cache RAM in use; represents the size of the proxy cache currently in use
AVE+60	CachedFiles	4	binary	number of proxy cached files; represents the number of files currently in the proxy cache
AVE+64	GETrequests	4	binary	number of GET requests; cumulative from Web server startup
AVE+68	HEADrequests	4	binary	number of HEAD requests; cumulative from Web server startup
AVE+72	POSTrequests	4	binary	number of POST requests; cumulative from Web server startup
AVE+76	CGIrequests	4	binary	number of CGI requests; cumulative from Web server startup
AVE+80	GWAPIrequests	4	binary	number of GWAPI requests; cumulative from Web server startup
AVE+84	Level200responses	4	binary	number of Error Level 200 (Error 200-299) responses; cumulative from Web server startup
AVE+88	Level300responses	4	binary	number of Error Level 300 (Error 300-399) responses; cumulative from Web server startup
AVE+92	Level400responses	4	binary	number of Error Level 400 (Error 400-499) responses; cumulative from Web server startup

Decimal Offsets	Name	Length	Format	Description
AVE+96	Level500responses	4	binary	number of Error Level 500 (Error 500-599) responses; cumulative from Web server startup
AVE+100	200responses	4	binary	number of Error 200 responses; cumulative from Web server startup
AVE+104	302responses	4	binary	number of Error 302 responses; cumulative from Web server startup
AVE+108	401responses	4	binary	number of Error 401 responses; cumulative from Web server startup
AVE+112	403responses	4	binary	number of Error 403 responses; cumulative from Web server startup
AVE+116	404responses	4	binary	number of Error 404 responses; cumulative from Web server startup
AVE+120	407responses	4	binary	number of Error 407 responses; cumulative from Web server startup
AVE+124	500responses	4	binary	number of Error 500 responses; cumulative from Web server startup
AVE+128	SMFRecordInterval	4	binary	interval since last SMF write in seconds
Note: Use the following fields only if the length of the SMF record type is greater than 192 bytes				
AVE+132	Diagnostic 1	4(signed)	binary	IBM internal use only
AVE+136	Diagnostic 2	4(signed)	binary	IBM internal use only
AVE+140	Diagnostic 3	4(signed)	binary	IBM internal use only
AVE+144	Diagnostic 4	4(signed)	binary	IBM internal use only
AVE+148	SMFConnectCnt	4	binary	number of connections since the last SMF record write; cumulative since the last SMF record was written
AVE+152	Reserved	4	binary	Reserved for future use
AVE+156	DNSMax	8	binary	maximum DNS lookup response time in seconds; value in C floating point (double) format; represents the longest response time since Web server startup
AVE+164	DNSMin	8	binary	minimum DNS lookup response time in seconds; value in C floating point (double) format; represents the shortest response time since Web server startup
AVE+172	DNSAvg	8	binary	average DNS lookup response time in seconds; value in C floating point (double) format; represents the average response time since Web server startup
AVE+180	ServicePluginsMax	8	binary	maximum service plugins response time in seconds; value in C floating point (double) format; represents the longest response time since Web server startup
AVE+188	ServicePluginsMin	8	binary	minimum service plugins response time in seconds; value in C floating point (double) format; represents the shortest response time since Web server startup

Decimal Offsets	Name	Length	Format	Description
AVE+196	ServicePluginsAvg	8	binary	average service plugins response time in seconds; value in C floating point (double) format; represents the average response time since Web server startup
AVE+204	CGIMax	8	binary	maximum CGI response time in seconds; value in C floating point (double) format; represents the longest response time since Web server startup
AVE+212	CGIMin	8	binary	minimum CGI response time in seconds; value in C floating point (double) format; represents the shortest response time since Web server startup
AVE+220	CGIAvg	8	binary	average CGI response time in seconds; value in C floating point (double) format; represents the average response time since Web server startup
AVE+228	SSLHandshakeMax	8	binary	maximum SSL handshake response time in seconds; value in C floating point (double) format; represents the longest response time since Web server startup
AVE+236	SSLHandshakeMin	8	binary	minimum SSL handshake response time in seconds; value in C floating point (double) format; represents the shortest response time since Web server startup
AVE+244	SSLHandshakeAvg	8	binary	average SSL handshake response time in seconds; value in C floating point (double) format; represents the average response time since Web server startup
AVE+252	ProxyResponseMax	8	binary	maximum proxy response time in seconds; value in C floating point (double) format; represents the longest response time since Web server startup
AVE+260	ProxyResponseMin	8	binary	minimum proxy response time in seconds; value in C floating point (double) format; represents the shortest response time since Web server startup
AVE+268	ProxyResponseAvg	8	binary	average proxy response time in seconds; value in C floating point (double) format; represents the average response time since Web server startup
AVE+276	SMFJobName	8	EBCDIC	z/OS job name; the job name may change if the Web server is restarted.
AVE+284	SMFJobASID	8	binary	z/OS address space ID where the Web server is running; this ID may change if the Web server is restarted.

z/OS console commands

The HTTP Server allows you to use the z/OS MODIFY and Workload Management console commands to display information on the Web server. For a description of command options, see Appendix A, “Commands” on page 383.

Chapter 14. Rating Web sites and serving rated Web information

PICS overview 285
Who can rate Web sites 285
How Web clients use PICS 286
How the HTTP Server helps you manage PICS labels 287
PICS for Web site administrators 287
Managing PICS labels for your Web site in each document 288
Managing PICS labels for your Web site from a central file 288
PICS for rating services and label bureaus . . . 288
How to manage PICS labels from a central file . . 289
Storing the PICS files on your server 290
Managing PICS labels for your own Web site 290
Starting a PICS rating service and label bureau 290
How to create PICS labels 291
PICS label extensions 291
How to request PICS label information 292
How to update the PICS configuration file . . . 293
Using the online Configuration and Administration forms. 293
Editing the PICS configuration file manually 293
Platform for Internet Content Selection configuration file syntax 294
Using wildcards in the Platform for Internet Content Selection configuration file 295

PICS overview

The Platform for Internet Content Selection (PICS) allows users of Internet applications, such as the World Wide Web, FTP, and Gopher, to filter the material they encounter, and accept or reject the material based on its ratings. This filtering allows parents, businesses, schools, or discerning individuals to block the access to inappropriate and objectionable material.

For the most up-to-date PICS information, see the World Wide Web Consortium's PICS Web site at URL:

`http://www.w3.org/PICS/`

The specifications published at this Web site enable:

- Content providers (people who publish information on the Web) to rate and label their own documents. These can be HTML files, or other files that contain images, sound, or animations.
- Independent rating services to rate and label documents published by other Web sites and to distribute the labels to whomever requests them.
- Internet users (browsers and other clients) to request these labels and determine how to handle rated and unrated information.

The HTTP Server makes it easy for you to store and serve the rating labels for the documents you publish. It also allows you to act as a rating service or label bureau by providing a means for you to maintain and distribute rating labels for other Web sites.

Who can rate Web sites

Web sites can rate themselves or be rated by a third party, called a ***rating service***. A rating service evaluates Web content according to their own published criteria and then distributes the labels through a ***label bureau***. Often a rating service acts as its own label bureau and distributes its own labels.

Some rating services will also give you assistance in assessing and labeling your own site and documents. The World Wide Web Consortium publishes a list of PICS self-rating services on its Web site at URL:

Rating Web sites and information

`http://www.w3.org/PICS/`

The PICS specification does not determine who can or will act as a rating service. The World Wide Web Consortium publishes a list of PICS third-party rating services on its Web site. In addition, anyone who wants to can set up a rating service. You can set up such a service by:

- Deciding on a rating system
- Publishing the rating system
- Rating documents and creating the rating labels
- Establishing a Web site (URL) that clients can access to get your labels

A rating service can choose any criteria on which to rate Web sites. While some might rate Web sites for their violence or sexual content, others could choose to rate educational content, political correctness, or even how "cool" the site is. Also, a rating service can rate any and all Web sites that it wants to rate.

Having your Web site and pages rated is often desirable. In fact, it may even be necessary for your Web site to be rated in order to be viewed by a PICS-enabled client. Understanding how Web clients use the PICS labels and ratings will help make this clear.

How Web clients use PICS

PICS-enabled clients allow the users to determine which rating services they want to use and, for each rating service, which ratings are acceptable and which are unacceptable.

For example, a family might choose a rating service that rates documents according to their sexual content. The rating service might have a low rating for romance, a higher rating for passionate kissing, and yet higher ratings for more explicit sexual activity. The parents might decide that documents containing romance are the highest acceptable rating for their household. They would then configure their browser to reject all documents that are unrated or contain a higher rating from this rating service.

In another example, the Hi-Tek Systems Corporation could label its own documents with a "For Hi-Tek Use Only" and could equip all its employees with browsers configured to accept only documents with that rating.

There are several steps in this process:

The client sends a request
When a PICS-enabled client requests a document, it indicates in the request which rating services are of interest. For example, assume these parents had configured their browser to evaluate rating labels from *The Best* rating service. When their children click a link to an HTML document, the browser request would also ask for the rating labels that were assigned to the document by this rating service.

The server sends a response
Assume the PICS-enabled server has a copy of the labels the client is requesting. When the server receives the client's request, it sends the labels along with the requested document. However, if the server does not support PICS or does not have copies of labels from that particular rating service, it sends the requested document anyway.

The client checks the server response first

The client first checks to see if the requested ratings labels are imbedded in the document (in the meta information) or if they were sent along with the document. Some clients might accept rating information that is imbedded in the file. Others might require a separate label from a registered rating service and a guarantee that it was created by that service. If the client successfully finds the label information it wanted, it evaluates the rating and either displays the document or blocks it and displays a message.

The client contacts the rating service, if necessary

If the client does not receive the label information with the requested document from the server, it might send a subsequent request directly to the rating service asking for the label information for that document. This requires a second connection, which takes longer and can discourage future visits to that site. The browser waits until the label information is returned before it displays **any** data.

Faster response time is the main reason why rating labels for a site should reside **at the site**.

How the HTTP Server helps you manage PICS labels

Whether your Web server publishes Web documents or you are a rating service and want to provide the labels for other Web sites, the HTTP Server can help you manage PICS labels.

Note: If you are going to use your server to rate your own documents or to run a label bureau, we strongly suggest that you use the default server port (80).

PICS for Web site administrators

As more browsers are configured to block access to unrated documents, it behooves you to have your Web site rated. And because it saves time when a browser can get the ratings when it sends its initial request, it behooves you to store the ratings for your pages on your own server. With the HTTP Server's PICS support, you can manage the labels from one central file and serve them with requested pages and documents. These labels can be:

- Self-assessed according to your own criteria

  If you are establishing your own rating service, you can rate your own site according to your published criteria.
- Self-assessed according to the published criteria of a voluntary rating service

  Voluntary rating services, such as SafeSurf (http://www.safesurf.com) trust Web administrators to be honest in the assessment of their own pages.
- Assessed by a third-party rating service according to the service's criteria

  In this case, you might contact the rating service and request that they rate your Web site (if they have not already done so) and send you the label information. In fact, you might want to contact several rating services to have your site rated for different subject criteria. If the third-party rating services have the HTTP Server, this process can be simplified with an electronic request. See "How to request PICS label information" on page 292.

Once the ratings are established, Web administrators can do one of three things:

- Manually edit each of their HTML files, inserting the rating information in the headers. See "Managing PICS labels for your Web site in each document" on page 288.

- Use the label information to create PICS-compliant rating labels, store the labels in their file system, and use the PICS configuration file to manage and transmit them. See "Managing PICS labels for your Web site from a central file".
- Let the system automatically store the transmitted rating labels and update your PICS configuration file for you. This can only be done when electronically requesting labels for a third-party rating service that has the HTTP Server. See "How to request PICS label information" on page 292.

Managing PICS labels for your Web site in each document

You can edit each of your HTML files and embed PICS ratings information in the meta element of the document header. This process is entirely manual and therefore time-consuming, error-prone, and difficult to maintain. It does not incorporate any of the security mechanisms (message digest, digital signature, etc.) that would guarantee the validity of the label, if this is important to the requesting client. The PICS specification explains how you can embed rating information in each document. It is not covered here.

To access the PICS specification, go to URL:

`http://www.w3.org/PICS/`

Managing PICS labels for your Web site from a central file

The HTTP Server's PICS support allows you to store the rating labels for all the documents on your Web site and manage them from a central file. The labels are sent along with your Web pages when a client requests them.

In addition to the rating labels, you must also have a PICS-compliant rating system description file that describes the rating system used to rate your documents. These are called RAT files, and rating services will provide them along with their labels.

Once you have both the labels and the RAT file, you can use the PICS configuration file to manage these labels from a central point. See "How to manage PICS labels from a central file" on page 289.

PICS for rating services and label bureaus

Because many Webmasters will want their pages rated, you have an opportunity to provide a service to a large number of Web sites.

- Content providers will contact your organization to request that you rate their Web site and provide them with the labels so that they will be able to serve the labels along with their Web documents themselves.
- Clients will connect to your server electronically to request labels for pages they are attempting to view only when they ***cannot*** get the label information with the requested pages.

The PICS configuration file provides you with the means to manage the labels for other Web sites and transmit them when requested.

The PICS specifications allow anyone to set up a rating service, define the criteria by which they rate Web sites and documents, and then provide the ratings. With PICS support, you can establish your server as a rating service and maintain and distribute labels for other Web sites. You can rate documents at a Web site individually or use wildcard characters to quickly assign the same rating to all or part of a Web site's offerings. You will need to create these labels and your own RAT file. The RAT file is a PICS-compliant rating file that describes the rating

system used to rate documents. Once you have both the labels and the RAT file, you can use the PICS configuration file to manage these labels from a central point. Your server will then be able to automatically send the rating labels you have assigned when a client requests them. See "How to manage PICS labels from a central file".

If a Web site that you have rated requests the labels for their pages, you can also provide them with all their current ratings. Unfortunately, the World Wide Web Consortium has not yet defined a standard for the label bureaus or rating services to send a Web site all their label information. This means that the method for this exchange will have to be determined by the rating services and the Web sites that ask for them.

If the Web sites and the rating service (or label bureau) both have the HTTP Server, they can electronically exchange rating labels and label entries for their PICS configuration file. In this case, the rating labels will be automatically stored on the server and the PICS configuration file will be updated so that it can transmit the labels with the requested documents.

If not, we are assuming that the rating services will send a file of all the required label information to the Web site administrators. Once the administrators receive this information, they will use whatever method is available on their server to create PICS rating labels and enable their server to transmit them with the requested documents.

How to manage PICS labels from a central file

Managing PICS labels requires three things:

1. A rating (RAT) file that describes the ratings

   If you are starting your own rating service or label bureau, you will need to create a file that describes your rating system. This file must be in the machine-readable format detailed in the PICS technical specifications and it should have the **.rat** extension. If you are getting your labels from a third-party rating service, you must also get a copy of their RAT file.

2. The rating labels themselves

   Whether you are maintaining labels for your own Web site or, as a rating service, maintaining labels for other sites, you will need to store the labels in your server's file system, one label per file. Rating services will rate documents on the Internet and create the rating label files themselves. Web sites will either rate their own site and create the label files or they will request the rating labels from third-party rating services.

   If you are getting your rating labels from a third-party rating service that also has the HTTP Server, you can request the labels electronically. They will be sent and stored directly on your system for you. Otherwise, you may need to do some editing of the information you receive before creating rating labels to store in your file system.

   When creating PICS rating labels, be sure to follow the PICS specification. See "How to create PICS labels" on page 291. We recommend you use **.lbl** for the extension on your label files. We have included a predefined AddType directive in the configuration file for this extension.

3. The PICS configuration file

   This file provides a mapping between the actual rating labels and the documents they rate. It enables the server to quickly respond to HTTP, FTP, and Gopher requests. If you are getting your rating labels from a third-party

rating service that also has the HTTP Server, your PICS configuration file will automatically be updated with entries for the labels you receive. If you are a rating service or if you receive rating labels from third-party rating services that have a different server, you will need to maintain the PICS configuration file yourself. You can use the online Configuration and Administration Forms to update and maintain this file or you can edit it manually. See "How to update the PICS configuration file" on page 293.

Storing the PICS files on your server

You will need to store both the RAT file and rating labels in files on your server.

The RAT file should be available from a rating service's Web site. The rating labels must be stored one label per file.

You can use any directories, subdirectories, and file names that make sense at your site and for your implementation. We recommend that Web sites have a separate directory or subdirectory for each third-party rating service that they use. This is **required** for automatic updates when requesting labels from rating services that have the HTTP Server.

Our examples use a file extension of ***.lbl*** on each rating label file. This is also the extension for any label files the server transmits electronically.

Managing PICS labels for your own Web site

Follow these steps to store rating labels in your file system and configure your server so it sends these labels when clients request them.

1. Obtain a copy of the RAT file from the rating services you want to use and store it in your file system on your server.
2. If you are getting rating labels from a third-party rating service that has the HTTP Server:
   - Use the online Configuration and Administration Forms to request the labels and the entries for your PICS configuration file electronically. When you receive these files, your server will automatically be updated for you. See "How to request PICS label information" on page 292.

   If you are **not** getting rating labels from a third-party rating service that has the HTTP Server:
   - Obtain the ratings from the third-party rating service or rate your own documents.
   - Create labels according to the format published in the PICS specification. See "How to create PICS labels" on page 291.
   - Store the labels in separate files, one label per file, in your server's file system.
   - Tell your server which documents are rated, where the actual rating labels can be found, and which rating service provided the labels. You do this by adding entries to the PICS configuration file to associate the rated documents with their label files. You can use the online Configuration and Administration Forms to update and maintain this file or you can edit it manually. See "How to update the PICS configuration file" on page 293.

Starting a PICS rating service and label bureau

Follow these steps to configure your server as a PICS rating service, store rating labels for other Web sites, and serve them in response to client requests.

1. Define a rating system and create your own RAT file. Check the World Wide Web Consortium's PICS specification for instructions on how to do this. It includes the syntax for the machine-readable format of the RAT file.

   You can access the PICS specification at URL:

   `http://www.w3.org/PICS/`

2. Establish two URLs for your rating service. One URL identifies your service by name. Include this URL in your RAT file. The other URL is for label requests. You must direct all the label requests that come to your server to this specific URL.

   The PICS specification has no requirements regarding these URLs; you may choose any URL that you like.

   Add the `Service` directive to the configuration file to inform the server that you are a PICS service and specify where to direct the PICS rating label requests. To add the Service directive, use the Request Routing form in the Configuration and Administration forms. For example:

   ```
   Service /Ratings INTERNAL:PICS-Ratings
   ```

   Replace */Ratings* with the path and file name portion of the URL you will use for label requests. For example, if you publish the URL http://www.coolratings.com/CoolSite, you would only include /CoolSite in the Service directive.

   Note: It is advisable to use the Configuration and Administration form, Request Routing, to ensure correct placement of the Service directive in the configuration file. If the Service directive is inserted manually, place it with the default Service directives defined in the configuration file.

3. Rate documents and Web sites according to your established rating system.
4. Create rating labels for these documents and sites and store them in your server's file system, one label per file. See "How to create PICS labels".
5. Tell your server which documents you have rated, what host serves them, and where the labels can be found in your file system. You do this by putting entries in the PICS configuration file that associates the rated documents and their specific label files. You can use the online Configuration and Administration Forms to update and maintain this file or you can edit it manually. See "How to update the PICS configuration file" on page 293.
6. Make the URL you will use for label requests known to the public.

   Notify all your subscribers and users to send their requests for rating labels to this URL. PICS-enabled clients and servers will use this URL to contact your server for labels.

How to create PICS labels

In general, a label file is a text file containing a label. Carefully review the format of labels given by the PICS Rating Services and Rating Systems specification.

You can access the PICS specification at URL:

`http://www.w3.org/PICS/`

PICS label extensions

The HTTP Server has added extensions to this format to save you repetitious data entry and to allow you to add comments.

Comments for your own use
You can insert comments for your own use into label files. Begin these comment lines with '#'. Lines beginning with '#' are not sent to clients. This type of comment is an addition to the "comment" statements used inside labels. "Comment" statements in labels are sent to clients.

Additional variables
You can insert some variables in label files:

%%URL%%
The current URL will be substituted for this variable. When the server receives a request for a rating label that contains `for %%URL%%`, it replaces this variable with the correct `for` statement before sending the label.

Note: Do not use this variable on generic labels (those that apply to multiple files).

%%SERVICENAME%%
The service name requested will be substituted for this variable. When the server receives a request for a rating label that contains `for %%SERVICENAME%%`, it replaces this variable with the correct service statement before sending the label.

How to request PICS label information

If a third-party rating service has the HTTP Server, you can electronically request rating labels for all the documents on your Web site that the third-party service has rated. As a response to that request, you will receive both rating labels and PICS configuration file label entries. Both types of information will automatically be stored on your server.

To electronically request rating label and entries for automatic update:

1. From the default home page (Frntpage.html), select **Configuration and Administration Forms**. When prompted, enter the administration user ID and password you have set up.
2. Select **PICS Services Configuration**. This displays the PICS Services Configuration main page.
3. Select **Request Label Entries from Third-Party Rating Service**.

Note: The third-party rating service **must** have the HTTP Server for you to use this feature. If not, the request fails.

If the third-party rating service has rated your Web site, it will return both the rating labels and label entries for your PICS configuration file. The rating labels will be stored in the directory you specified on the form. The label entries will automatically be added to your PICS configuration file.

If the third-party rating service has not rated your Web site, it will return a response indicating that it does not have the information you requested.

How to update the PICS configuration file

The HTTP Server provides the PICS configuration file for you to manage PICS labels from a central point and serve them when clients request them. You can use the online Configuration and Administration forms to add, modify, and delete the label entries in the PICS configuration file, or you can edit the file and maintain the data manually.

Using the online Configuration and Administration forms

1. From the default home page (Frntpage.html), select **Configuration and Administration Forms**. When prompted, enter the administration user ID and password you have set up.
2. Select **PICS Services Configuration**. This displays the PICS Services Configuration main page.
3. If you are maintaining labels for your own Web site:
   a. Select **Register Third-Party Rating Services** to register the services that have sent you labels and identify their RAT files. With the PICS example files, initially you will have one entry for the The Best rating service, **http://www.coolness.raleigh.ibm.com/ratings/V1.html**, along with its RAT file, coolness.rat.
   b. Select **Maintain PICS Label Entries for Your Web Site** to view, add, modify, or delete the entries that associate specific documents or pages with your rating labels.

   If you are maintaining labels for other Web sites:
   a. Ensure that you have your RAT file stored in your file system.
   b. Select **Register Your Own Rating Service** to register the location of your RAT file on your server.
   c. Select **Maintain PICS Label Entries for Other Web Sites** to view, add, modify, or delete the entries that associate specific documents or pages with your rating labels.

Editing the PICS configuration file manually

The exact name and location of the PICS configuration file is etc/ics_pics.conf.

The configuration file consists of a list of paragraphs. There are three types of paragraphs.

- **LabelsFor**

  Specifies the ratings given by a particular rating service for documents on a given Web server. For example, one LabelsFor paragraph could cover ratings according to the RSAC rating system for documents on the local server, while another paragraph could cover ratings according to the Best rating system for documents on the local server.
- **DefineService**

  Lists local label files associated with a third-party rating service.
- **DefineLBService**

  Lists local label files associated with your own label bureau or rating service.

Note: The PICS configuration file associates Web documents with files containing labels. The labels themselves are stored in separate files, not in the PICS configuration file.

Platform for Internet Content Selection configuration file syntax

LabelsFor:

The first line of the paragraph consists of the keyword `LabelsFor`, the name of the server on which the rated documents are found, the name of the rating service, and an opening brace. The body of the paragraph specifies labels for sets of documents. Each paragraph ends with a closing brace.

```
LabelsFor servername servicename {
    /WebPath1/document1    /path/LabelFile1
    /WebPath2/document2    /path/LabelFile2
    ...and so on...
}
```

servername
: This can be the keyword `LOCAL` to indicate documents on this server, or it can be a full URL if documents on remote servers are being rated. Only servers acting as label bureaus (rating services) will need to use a hostname other than `LOCAL`. When your server is providing labels for the documents it hosts, you should always use the keyword `LOCAL` for the hostname. Note that you must specify the protocol and hostname without a trailing slash; thus, `http://www.xyz.com` is acceptable as a hostname on a LabelsFor line, but `http://www.xyz.com/` is not.

servicename
: The full URL where clients will send their label requests.

/WebPath/document
: The Web path and name of the document being rated. This is the path a Web client would use when requesting the document. For example, if the Naughty/Image1.gif was on the server `www.rated.xyz.com`, then a Web client would request `http://www.rated.xyz.com/Naughty/Image1.gif`.

Note: You can use wildcard characters (*) to rate multiple documents at once. See "Using wildcards in the Platform for Internet Content Selection configuration file" on page 295.

/path/LabelFile
: The fully qualified name of the label file in your file system.

You cannot use wildcard characters in file names.

A special keyword, `NOTLABELED`, can be used in place of a label file name. This indicates that the given file(s) cannot be labeled; it serves as a shorthand way of creating a label file that contains a "not-rated" label. In the example above, a not-rated error message will be returned to any clients who request a rating for the file /Unknown.html.

For example, an actual LabelsFor paragraph might look like this:

```
LabelsFor LOCAL http://www.rsac.org/ratingsv01.html {
 /Naughty/Image1.gif  /usr/lpp/internet/server_root/labels/AdultsOnly.lbl
 /Clean/*.html        /usr/lpp/internet/server_root/labels/AllAges.lbl
 /Unknown.html        NOTLABELED
}
```

DefineService:

The first line of the paragraph consists of the keyword `DefineService`, the rating service URL, the quoted name of the rating service, the location and name of the service's RAT file, and an opening brace. The body of the paragraph lists the label

files associated with this service, specifying each one with the keyword `LABELFILE`. Each paragraph ends with a closing brace.

```
DefineService servicename "name-of-service" ratingfile {
     LABELFILE  /path/LabelFile1 "description"
     LABELFILE  /path/LabelFile2 "description"
     ...and so on...
}
```

servicename
: The name (URL) of the rating service.

name-of-service
: The name (text) of the rating service, in quotes.

ratingfile
: The fully qualified name of the service's RAT file in your file system.

/path/LabelFile
: The fully qualified name of the label file in your file system.

description
: A text description of the label, in quotes.

For example, an actual DefineService paragraph might look like this:

```
DefineService http://www.abc.org/rate.html "The ABC's of
     Ratings" d:\www\pics\rat\abc.rat {
   LABELFILE  /usr/lpp/internet/server_root/AdultsOnly.lbl "rated XXX"
   LABELFILE  /usr/lpp/internet/server_root/AllAges.lbl    "rated GGG"
```

Note: In this example the DefineService statement ending with the left brace appears on two lines for printing purposes. This statement appears on one line in the Platform for Internet Content Selection (PICS) configuration file.

DefineLBService:

This paragraph has the same syntax and format as the `DefineService` paragraph. The only difference is that it uses the `DefineLBService` keyword. The RAT file and labels that it lists are for your own label bureau and rating service.

Using wildcards in the Platform for Internet Content Selection configuration file

You can use an asterisk (*) as a wildcard only in the LabelsFor paragraphs of the PICS configuration file. When using wildcards, remember that the order of entries within a paragraph is important. For each paragraph, the HTTP Server breaks the list of rated documents into two parts: those that contain wildcards, and those that do not contain wildcards.

- When the server looks for labels for a document, it will first try to find the document in the "no-wildcards" list. Order is **unimportant** here. Without wildcards, each entry in the list refers to exactly one document.
- If the server cannot find a match in the "no-wildcards" list, it will try to match the document name against the entries that contain wildcards. Order is **important** here. The server tries to match the requested document against the wildcard entries in the order in which they appear in the configuration file and will use the first entry that matches.

For example, if you want an entry that gives `/*` as the WebPath/document, serving as a catchall for documents that don't have another rating, then make this the last entry in the paragraph.

Chapter 15. Retrieving Lightweight Directory Access Protocol information

Querying the LDAP server 297
LDAP search filters 297
Examples of LDAP search filters 297
Configuring LDAP on the HTTP Server 298
Using LDAP to protect files 300
Using LDAP to share configuration information 300
To manually search a directory server . . . 301
To add an entry 301
To modify an existing entry 301
Determining why the Web server cannot establish a connection to or bind to Lightweight Directory Access Protocol 302
Lightweight Directory Access Protocol in a Secure Sockets Layer environment 302

Querying the LDAP server

LDAP search filters

LDAP accesses the X.500 directory through the use of human readable strings. When these query strings are passed to the LDAP server, the server returns the distinguished name of the entry.

LDAP entries are typed, or classified, by an ObjectClass attribute to simplify searches. For example, you could search an LDAP directory whose objectclass=acl to locate all entries that are access control lists.

A search filter for an LDAP entry has the following structure:

- Filters must begin and end with parentheses. See the following examples which show the placement of parentheses in complex queries.
- Filters may contain the following Boolean comparisons:
  - & - Boolean AND
  - | - Boolean OR
  - ! - Boolean NOT
- Depending on the objectclass, an attribute name may be required.
- Filters may contain the following equality expressions
  - = - equal to
  - ~= - approximately equal to
  - >= - greater
  - <= - less
- Filters must contain the value of the attribute to search on. This may contain wildcards.

For more information on LDAP search filters, see RFC 1960.

Examples of LDAP search filters

(cn=Joe Smith)
searches the directory service for the common name of Joe Smith. Possible matches are:

```
Joe Smith
```

(!(cn=Jane Doe))
queries the directory service for entries whose common name is not Jane Doe. Possible matches are:

```
Joe Schmoe
Adam Fosset
any name other than Jane Doe
```

(&objectClass=acl)((sn=Johnson)
queries all acl entries matching a surname of Johnson. Possible matches are:

```
Peter Johnson
Davey Johnson
```

(o=univ*of*carolin*)
queries the organization attribute. Possible matches are:

```
University of North Carolina Chapel Hill
University of South Carolina
```

Note: LDAP can return more than one entry. However, the Web server will not authenticate when multiple entries are returned. If the University of North Carolina and the University of South Carolina were included in the directory queried by this example, both would be returned, and the authentication would fail. The search filter must be altered.

Configuring LDAP on the HTTP Server

Current limitations:

- If the LDAP server does not allow writing via the LDAP client API (as in the case of the Lotus Notes server), the HTTP Server cannot store configuration information using the ldapadd command in the LDAP client toolkit.
- There is a known problem with storing and retrieving configuration information on the Telstra server.

Using LDAP to protect files

This section will show you how to protect files or directories using user or group information on an LDAP server.

1. From the server Front Page, click **Configuration and Administration Forms**.
2. Click **Access Control**, then **Document Protection**.
3. Fill in the following fields:
   - URI request template - URI template of files or directories to protect. Wildcard is accepted.
   - Select the position in the list.
   - Select whether you want the protection made in-line or as a named protection.
   - To share access control on an external shared LDAP server, select the **Named LDAP setup** option.

     Select the existing LDAP setup from the list or select **Create a new LDAP setup...**
   - Optionally, enter the IP address or host name of the requesting machine.
4. Click **Submit** to continue, or **Reset** to clear the form.
5. To create a new LDAP connection, you must provide information about the LDAP server being used. This creates an LDAPInfo directive. See “LDAPInfo - Define an external LDAP server” on page 485 for more information on this directive.

   Enter the following general LDAP settings:

- LDAP Setup Name - name of the LDAP server label associated with a group of LDAP parameters. For example, *PrimaryLdapServer*
- Protection Realm - name of the protected area as seen by the requesting client. For example, *Administrator Access*
- Permissions - enter the names of the users or groups which have read, write, or delete permissions. The name or group must be a valid distinguished name. For example, *cn=group1, o=IBM, c=US*
- Host Name - hostname of the LDAP server. For example, *ldap.ibm.com*
- Connect via - transport method used. Values include TCP or SSL
- Port Number - optional port number on which the LDAP server is listening. The default for TCP connections is 389 and 636 for SSL
- Key file name - file name of the key file database. This is required if you are using SSL.
- Key Label - name of the certificate label the Webserver uses to authenticate with the LDAP server. This label is required if using SSL as the transport. For example, *My Server's Certificate*

6. Enter the Web server connection information:
   - Server Authentication Type - specify the method for authenticating the Web server to the LDAP server. Possibilities are:
     - None - if the LDAP server does not require the Web server to authenticate.
     - Basic - the HTTP Server must provide a userid and password to access the information on the LDAP server. The Web server's distinguished name is used as the userid, and the password stored in the stash file is the password.
   - Server's Distinguished Name - distinguished name of the Web server. This name is used as the username when accessing an LDAP server using Basic authentication.
   - Server Password - password for the server to log on to the LDAP server. This is required only if Server Authentication type is Basic. The password will be encrypted and stored in a stash file named in the `ServerPasswordStashFile` subdirective.
7. Enter client connection information:
   - User Authentication Type - authentication type for connections made by the client
   - User Search Base - starting point for the LDAP server to search user names
   - User Name Filter - filter used to convert the username as input by the user to a search filter for an LDAP entry. The default is `"(&(objectclass=person)(cn=%v1* %v2*))"` where %v1 and %v2 are the words typed by the user.

     For example, if the user types `"Pa Kel"`, the resulting search filter would be `"(cn=Pa* Kel*)"`. Search filter syntax is described in "LDAP search filters" on page 297.

     However, because the Web server cannot differentiate between multiple returned entries, authentication fails when the LDAP server returns more than one entry. For example, if there are entries `(cn=Paul Kelsey)` AND `(cn=Paula Kelly)` using the search filter above, the authentication will fail. You must modify your search filter.
   - User Name field separator - specify the set of characters used to delimit the users data. The default characters are the space, comma, and the tab (\t) character.

- User Certificate Filter - converts the information in the client certificate passed over SSL to a search filter for an LDAP entry. The default is `"(|&(objectclass=person)(cn=%v1)(ou=%v2)(o=%v3)(c=%v4))"`. SSL certificates include the following fields, all of which can be converted to a search filter:

Certificate field	Variable
common name	`%v1`
organizational unit	`%v2`
organization	`%v3`
country	`%v4`
locality	`%v5`
state or country	`%v6`
serial number	`%v7`

Note: When the search filter is generated, the values in the fields are placed into the matching variable fields (%v1, %v2). The following table shows the conversion:

Table 4. User Certificate Filter conversion

Certificate:	cn=Road Runner o=Acme Inc c=US
Filter	(cn=%v1, o=%v3, c=%v4)
Resulting Query	(cn=RoadRunner, o=Acme, Inc, c=US)

- Group Search Base - starting point for the LDAP server to search for group entries
- Group Name Filter - filter LDAP uses to search for group names. The default is `(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)`
- Group Member Attributes - specify the names of the +attributes which contain group member information. More than one attribute may be used to contain member information. The default attributes are member and uniqueMember.

8. Enter timeout settings:
   - Search Time-out - time limit an LDAP server is given to complete a search
   - Cache Time-out - amount of time a response returned from the LDAP server remains valid
   - Idle Connection Time-out - time limit before an idle LDAP server connection is closed due to inactivity
   - Connection retry interval - when a connection must be reestablished because of a down server, this is the time the Web server waits before attempting to reconnect
9. Click **Submit** to continue or **Reset** to clear the form.

Note: If you are using SSL client certificates in order to do authentication with LDAP, see "Creating protection setups for Secure Sockets Layer (SSL) client authentication" on page 168 for more information.

Using LDAP to share configuration information

To share configuration information, you must store it on an LDAP server. Storing information on an LDAP server differs from storing information on other systems.

Storing information on an LDAP server is usually done by an administrator at the LDAP server (see your LDAP server documentation for more information). The LDAP client toolkit, included in S/390, is used to access the X.500 directory service. Refer to the toolkit documentation for syntax.

To view LDAP documentation, go to the z/OS Internet Library at URL:

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

Other LDAP functions are described in Chapter 20, "Accessing LDAP information with the LDAP API" on page 379.

Note: The following examples use the applicationProcess objectclass to store configuration information. Although using the applicationProcess objectclass is not required, it is recommended.

To manually search a directory server

Use the following command to search an LDAP server for a list of applicationProcess objects:

```
ldapsearch -b "o=Ace Industry, c=US" -h <hostname>\
     "(objectclass=applicationProcess)"
```

Change ″o=Ace Industry, c=US″ to a search base which is valid for your LDAP server.

To add an entry

This example creates a new applicationProcess entry on the LDAP server. It assumes ″o=Ace Industry, c=US″ is a valid search base for your LDAP server, and that ″Directory Manager″ has the permission to add a new entry.

1. Create a text file named ″new″ containing:

   ```
   dn: cn=Go Webserver, o=Ace Industry, c=US
   objectclass: applicationProcess
   cn: HTTP Server
   ```

2. Enter the following command:

   ```
   ldapadd -f new -D "cn=Directory Manager, o=Ace Industry, c=US"\
        -w <password> -h <host>
   ```

To modify an existing entry

This example adds configuration information stored in a local file to the LDAP server directory. You can modify an entry as often as you need.

1. Modify the ″new″ file to contain:

   ```
   dn: cn=HTTP Server, o=Ace Industry, c=US
   objectclass: applicationProcess
   description: <path>
   ```

   where `<path>` is the full path name of a file containing configuration information to store in the LDAP directory.

   Note: Your original httpd.conf configuration file must contain an LDAPInfo and LDAPInclude directives. All other directives can be shared on the LDAP server.

2. Enter the following command:

   ```
   ldapmodify -b -f new \
        -D "cn=Directory Manager, o=Ace Industry, c=US"\
        -w <password> -h <host>
   ```

Determining why the Web server cannot establish a connection to or bind to Lightweight Directory Access Protocol

Under certain conditions, such as security violations, the Web server is unable to give an explanation of why it cannot establish a connection to or bind to the LDAP server. You may be able to diagnose the problem by adding the following statement to the Web server `httpd.envvars` file:

```
LDAP_DEBUG=65535
```

The statement causes the z/OS LDAP client code to write messages to standard error. Be sure to code the DD statement for standard error so that you capture the messages. If you have one of the Web server tracing options turned on, the messages from the LDAP client code intersperse with the Web server trace messages.

The messages from the LDAP client code are undocumented because they are internal to the LDAP client code. However, you can find the messages helpful. The LDAP server does not have to reside on a z/OS machine for the LDAP client code to capture the messages.

Lightweight Directory Access Protocol in a Secure Sockets Layer environment

The Web server can establish Secure Sockets Layer (SSL) connections to Lightweight Directory Access Protocol (LDAP) servers and to clients. When the Web server connects to an LDAP server, it calls the LDAP client code. Both the Web server and the LDAP client code are capable of initializing SSL. However, the SSL code can only initialize once. The Web server does the SSL initialization and does not allow the LDAP client code to initialize SSL.

If you write code that manages connections to LDAP servers, your code is under the same restriction as the LDAP client code. Your code cannot initialize SSL. If both your code and the Web server code initialize SSL, SSL connections between the Web server and the LDAP server work improperly. Before you call the ldap_ssl_start function, the ldap_ssl_client_init function, or the ldap_ssl_init function, call the ldap_set_option function with the following parameters:

```
ldap_set_option (NULL, LDAP_OPT_SSL_AVOIDSTATICINIT, NULL)
```

For information on LDAP functions, see *z/OS SecureWay® Security Server LDAP Client Programming*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Chapter 16. Running your server as a proxy

Web Traffic Express proxy features 303
Definitions 303
Setting up your proxy server 304
Guidelines for the *IBM Web Traffic Express for Multiplatforms User's Guide* 304
Setting up logging for your proxy server . . . 305
Setting up a forward proxy server 305
Setting up a reverse or hidden proxy server . . 306
Turning off your proxy server 307
Understanding the CACHEAGT program 307

Web Traffic Express proxy features

The HTTP Server includes the proxy, caching, and filtering features of IBM Web Traffic Express.

With these features, your server can act as a proxy, and retrieve Internet data from multiple servers. With the optional caching features, you can manage its caching functions to optimize server performance and minimize user response time. You can also filter the content you serve using Platform for Internet Content Selection (PICS) labels.

Web Traffic Express is also a socks-enabled proxy. It includes a flexible-client SOCKS feature that allows requests for specific IP addresses to go directly to the destination server instead of routing them through a SOCKS server.

Definitions

This section defines the main terms used when working with proxy servers.

proxy server
A server that accepts requests from clients (usually browsers, but possibly servers) and forwards these requests to servers. The proxy server can cache files that it receives from destination servers. The proxy server can then return these cached files to clients on subsequent requests, without requesting the files from the destination servers again.

origin server
The Web server that holds the original copy of the resource.

content server
Another term for origin server.

destination server
Another term for origin server.

firewall
A functional unit that protects and controls the connection of one network to other networks. The firewall prevents unwanted or unauthorized communication traffic from entering the protected network and allows only selected communication traffic to leave the protected network. The firewall can contain a proxy server.

socks server
A circuit-level proxy server which establishes a connection from a client to an application, and then forwards the data in both directions without further interference. This activity is not a function of Web proxy servers, but special purpose hardware and software.

forward proxy server
A proxy server configuration that requires users to define the proxy server in the users' browsers. An organization for a specific group of people usually sets up this type of proxy to handle requests on behalf of those people.

reverse proxy server
A proxy server configuration transparent to users because the users do not configure their browsers to point to the proxy.

hidden proxy server
Another term for reverse proxy server.

Secure Sockets Layer (SSL) tunneling
A proxy server forwards an SSL request to a destination server without decrypting the request. The tunneling proxy server only understands the destination server address and port number. The proxy server does not need SSL configured. Only the destination server needs to support SSL and decrypt requests. HTTPS requests come into the proxy server non-SSL port, which is port 80, by default. The proxy server redirects the requests to the destination server SSL port, usually set to 443. The SSL tunneling proxy server must have the CONNECT method set on the Enable directive. Most filtering by the proxy server becomes impossible when using tunneling. For example, virus screening is not possible. Filtering based on the URL or header fields also becomes impossible because the proxy server with SSL tunneling enabled does not decrypt this information. SSL tunneling only works for a forward proxy.

For a reverse proxy, you can implement an alternative to SSL tunneling. In this case the SSL connection only occurs between the browser and the proxy server. The connection between the proxy server and the origin server is non-SSL. The reverse proxy decrypts the information and passes it to the origin server. Use this implementation when the origin server lacks SSL capability and needs the proxy server to provide secure connections across the Internet or an intranet.

Setting up your proxy server

Use the following resources to set up your proxy server:

- Proxy server configuration directives described in "Proxy server settings — Configure the Web server as a proxy server" on page 529.
- *IBM Web Traffic Express for Multiplatforms User's Guide*. You can access this book on the Web site at URL:

  `http://www.ibm.com/software/webservers/httpservers/doc53.html`
- Topics in this chapter.

Guidelines for the *IBM Web Traffic Express for Multiplatforms User's Guide*

Use the following guidelines when referring to the *IBM Web Traffic Express for Multiplatforms User's Guide*:

- In general, use the examples for the AIX operating system.
- Skip the installation instructions. Web Traffic Express features automatically install when you install the HTTP Server.
- Use the instructions in the Configuration Quick Start chapter to enable basic proxy and caching functions and to set up a secure connection.

Note: By default the proxy function is disabled.

- Use the instructions in the remainder of the book to:
  - Manage the cache functions
  - Complete the automatic cache refresh configuration and start the cache agent
  - Configure PICS-based filtering
  - Complete the flexible-client SOCKS configuration
  - Set up SSL-tunneling
  - Customize header information
  - Use the Server Activity Monitor to review server activity and tune the server

If you have configured your proxy server as a forward proxy, your users need to configure their browsers to direct their requests to the proxy.

Setting up logging for your proxy server

You can optionally set up the following logging directives:

- ProxyAccessLog *path*
- CacheAccessLog *path*

where *path* can represent a full path or relative path. For more information on these directives, see "Proxy server settings — Configure the Web server as a proxy server" on page 529.

Setting up a forward proxy server

For a forward proxy, users must explicitly configure their browsers to direct their requests to the proxy server.

Set the following Proxy directives in the httpd.conf file for each protocol that you want to establish a proxy. You can set up your Proxy directives so that your Web server acts only as a proxy server, or as both a proxy server and an origin server. However, consider the security and control implications of mixing the two capabilities in the same Web server. When the Web server acts as a forward proxy server, the protocol specified in the browser is the same as the protocol that the origin server requires. However, for those requests in which the Web server acts as an origin server, you must specify HTTP as the protocol. Note that the Proxy directive takes only one argument when you define a forward proxy. You set the directive in the httpd.conf file.

HTTP protocol

```
Proxy http:*
```

FTP protocol

```
proxy ftp:*
```

When you access an FTP server through a proxy, use the following syntax for the URL:

```
ftp://[user ID[:password]@]host[:port]/path/file
```

Square brackets indicate optional parameters. Supply a user ID and password in the request if the FTP server requires a user ID and password.

Gopher protocol

```
Proxy gopher:*
```

SSL protocol with SSL tunneling

```
Proxy *:SSL_port_number
Enable CONNECT
```

For more information on the use of the Enable directive with the CONNECT method, see “Methods - Set method acceptance” on page 518.

Setting up a reverse or hidden proxy server

Users do not define their browsers to point to the proxy as they would in a forward proxy. The proxy is hidden from users. You can set up your Proxy directives so that your Web server acts only as a proxy server, or as both a proxy server and an origin server. However, consider the security and control implications of mixing the two capabilities in the same Web server. SSL or non-SSL requests can come into the proxy. The protocol specified in the browser is always HTTP, even if the Web server forwards the request to an origin server that is not an HTTP server. Note that the Proxy directive takes two arguments when you define a reverse proxy. You set the directive in the `httpd.conf` file. The format of the directive follows:

```
Proxy request-template URL/*
```

where:

request-template
: Represents a URI that you want your server to accept and to which you want the server to respond. You can use an asterisk as a wild card in the template.

URL
: Represents the address of the origin server that responds to the proxy request. It includes the protocol, followed by the host name or the IP address, and then the request. Specify the protocol as either FTP, gopher, HTTP, or HTTPS.

Note: When you specify an FTP server on *URL*, use the following syntax:

```
ftp://[user ID[:password]@]host[:port]/*
```

Square brackets indicate optional parameters. Supply a user ID and password if the FTP server requires a user ID and password.

Examples:

- `Proxy *  http://www.content.com:6780/*`

  In this case, the Web server only acts as a proxy, so it forwards all requests to the origin server.
- `Proxy /XYZ/*  http://www.destination.com/*`

  Since the Web server acts as a proxy for all URIs that start with `/XYZ/`, it forwards all requests that match the template to the origin server. For all other requests the Web server acts as an origin server.
- `Proxy /abc/*  ftp://user1:fred@www.ftpa.com/*`

  Since the Web server acts as a proxy for all URIs that start with /abc/, it forwards all requests that match the template to the FTP server. In this example the FTP server requires a user ID and password. Therefore, you supply a user ID and password as part of the URL on the Proxy directive. For all other requests the Web server acts as an origin server.

Turning off your proxy server

If you do not need proxy services, turn them off by commenting out the following directives in your Web server configuration file:

```
ServerInit            /usr/lpp/internet/bin/Jav_dll.so:Javelin_init
Service  /cgi-bin/dogc.icapi   /usr/lpp/internet/bin/Jav_dll.so:doGC
PreExit          /usr/lpp/internet/bin/Jav_dll.so:Javelin_preFilter
Enable ICSERRORLOG /usr/lpp/internet/bin/Jav_dll.so:Javelin_errorLog
```

Understanding the CACHEAGT program

The Web server's proxy caching program is CACHEAGT. This program is located in the *install-path*/bin directory. The Web server automatically runs the CACHEAGT program at midnight if you enable proxy capability and you set the AutoCacheRefresh directive to **On** in the javelin.conf file. The CACHEAGT program reads the CacheAccessLog file to refresh the contents of the proxy cache.

If you do not use the caching feature of the proxy, the CACHEAGT program does nothing while it runs. Prevent the Web server from launching this program by coding the following in the `javelin.conf` file:

```
AutoCacheRefresh Off
```

You can run the CACHEAGT program manually from the z/OS UNIX System Services command line. For information on the cacheagt command, see "cacheagt command" on page 383.

For a detailed discussion on proxy caching, see the *IBM Web Traffic Express for Multiplatforms User's Guide*. You can access this book on the Web site at URL:

`http://www.ibm.com/software/webservers/httpservers/doc53.html`

The CACHEAGT program reads the Web server `httpd.conf` file and looks for the following directives:

- Hostname
- Port
- ServerRoot
- ErrorLog
- CacheAccessLog
- Caching with the directive set to On

Missing or incorrect values can cause the CACHEAGT program to give incorrect output.

The CACHEAGT program uses the following configuration files:

- Proxy configuration file:
  /*path*/javelin.conf
- Socks configuration file:
  /*path*/socks.conf
- Web server configuration file:
  /*path*/httpd.conf

where *path* must be the same for all three files. The configuration file names must equal the names specified. If you have already named these configuration files something else, you can create a link between the file names you use and the file names required by the CACHEAGT program. For information on links, see *z/OS*

UNIX System Services User's Guide. To access this book on the Web, go to the z/OS Book Server Web site at URL:
`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

The CACHEAGT program uses the following log files, where *path* must be the same for all three files.

- Web server error log:
  `/path/http-errors`

  You define the error log on the ErrorLog directive.
- Proxy cache log:
  `/path/httpd-cache`

  You define the cache log on the CacheAccesslog directive.
- CACHEAGT error log:
  `/path/cacheAgent-errors`

  The CACHEAGT program creates the CACHEAGT error log in the same directory as the httpd-errors file.

The CACHEAGT program creates the following files:

- `<install_path><server_root>/pub/reports/javelin/cacheagt.html`

  The cacheagt.html file contains a cache report for you to view.
- `<install_path><server_root>/pub/reports/javelin/hitsfile.txt`

where *server_root* represents the path specified on the ServerRoot directive in the `httpd.conf` file.

Tracelog directive

The CACHEAGT program uses the tracelog directive to log status messages as an alternative to writing messages to standard error. The directive takes effect when encountered by the CACHEAGT program. Therefore, put the tracelog at the beginning of the `javelin.conf` file. If you use the command line version of the program with the -vv flag, the program writes additional messages to the log, when tracelog is set.

Syntax:
tracelog */path/file*

Initial configuration file setting:
None

Program default setting:
None

Example:
```
tracelog /usr/lpp/internet/logs/cacheagt.log
```

Chapter 17. Running your server with multiple IP addresses or virtual hosts

Requirements for using multiple IP addresses or virtual hosts 309
Multiple IP addresses. 309
Virtual hosts. 309
Setting up your Web server to use multiple IP addresses or virtual hosts 310
Welcome pages 310
Mapping rules 310
Access Control 311

Requirements for using multiple IP addresses or virtual hosts

You may want to use one Web server to provide Web sites for multiple customers. For example, you might have two customers (customerA and customerB), both of whom want to make information about their companies available on the Web. You might want to put both Web sites on the same machine if the expected number of requests for the information is not great enough to justify a separate machine for each customer.

With the HTTP Server, you can use multiple IP addresses, virtual hosts, or both to provide multiple Web sites on one server.

Multiple IP addresses

To use multiple IP addresses, your server must be installed on a machine with multiple network connections.

If you have only one network connection and run two instances of the server on the same machine, then only one server has the benefit of using the default port number. Requests to the other server would have to include a port number.

If your machine has two network connections, you can run just one instance of the server and assign each customer to a different IP address. For each IP address you would define a different host name. So customerA could be www.customerA.com on IP address 9.67.106.79 and customerB could be www.customerB.org on IP address 9.83.100.45. You could then configure the server to serve a different set of information depending on the IP address the request comes in on. Because the server can accept requests from the default port of each network connection, requests to either host name would not require a port number.

If your server has multiple IP addresses, you can associate a specific Secure Sockets Layer (SSL) key database label with each individual IP address. For more information, see “SSL support for multiple IP addresses” on page 62.

Virtual hosts

With virtual hosts, no additional hardware is required, and you can save IP addresses. However, clients must support HTTP 1.1 or HTTP 1.0 with 1.1 Extensions.

With virtual hosts, you can run just one instance of the server and assign each customer to a different host. In the domain name server, you define your hosts and associate them with the IP address of your server. You can then configure the server to serve a different set of information depending on the host for which the request is made. Requests do not require a port number.

Multiple IP addresses or virtual hosts

To take advantage of SSL support for multiple IP addresses (multiple virtual hosts), you must assign each host name a different IP address and then define a specific key database label for each IP address. For more information, see “SSL support for multiple IP addresses” on page 62.

Setting up your Web server to use multiple IP addresses or virtual hosts

Setting up your Web server to use multiple IP addresses or virtual hosts is very similar. For multiple IP addresses, you’ll need to specify the IP address a request comes in on and for virtual hosts, you’ll need to specify the host name for which a request is made.

You configure the server to serve different information for each customer by indicating that certain parts of your configuration apply only to requests coming in on certain addresses or for certain hosts. You can configure three key parts of your server so that requests are processed differently based on the IP address they come in on or the host name in the URL.

- Welcome pages
- Mapping rules
- Access control

Welcome pages

You can specify a different set of file names to use as welcome pages depending on the address a request comes in on or the host name in the URL. The file names you define as welcome pages determine how the server responds to requests that do not contain a file name.

For example, you might want to specify that homeA.html is a welcome page only for requests received on 9.67.106.79 or for hostA, and homeB.html is a welcome page only for requests received on an address 9.83.100.45 or for hostB.

From the Configuration and Administration forms page, you can configure your list of welcome pages by clicking on **Initial Page**. From the Initial Page form, click the help icon for information on defining welcome pages and how to associate a welcome page file name with an IP address or a host name.

Alternatively, if you are editing the configuration file, you can add an *IP-address* or *hostname* at the end of a Welcome directive to associate a welcome page file name with an IP address or a host name. For details, see the description of the Welcome directive in “Directories and Welcome Page - Set viewing options” on page 459.

Mapping rules

You can specify a different set of mapping rules for the server to use depending on the address a request comes in on or the host name in a URL. Mapping rules map a request to a physical file on the server and determine whether the server processes a request.

For example, you might want to specify that a request beginning /cgi-bin/ received on address 9.67.106.79 or for hostA is mapped to the /customerA/cgi/ directory, and the same request received on 9.83.100.45 or for hostB is mapped to the /customerB/cgi/ directory.

From the Configuration and Administration forms page, you can configure your mapping rules by clicking on **Request Routing**. From the Request Routing form, click the help icon for information on how to use mapping rules and how to associate a mapping rule with an IP address or a host name.

Alternatively, if you are editing the configuration file, you can add an *IP-address* or *hostname* at the end of Exec, Fail, Map, Pass, and Redirect directives to associate the directive with an IP address or a host name. For details, see the description of these directives in "Resource mapping - Redirect URLs" on page 536.

Access Control

You can activate different protection rules for a request based on the address the request comes in on or the host name in a URL. Protection rules are defined in protection setups and determine how your server controls access to files and programs.

For example, you might want to specify that a request beginning /cgi-bin/ received on address 9.67.106.79 or for hostA is protected by the rules in a protection setup named PROT-A and the same request received on 9.83.100.45 or for hostB is protected by the rules in a protection setup named PROT-B.

From the Configuration and Administration forms page, you can configure how protection is activated by clicking on **Document Protection**. From the Document Protection form, click the help icon for information on protecting documents and how to associate protection with an IP address or a host name.

Alternatively, if you are editing the configuration file, you can add an *IP-address* or *hostname* at the end of DefProt and Protect directives to associate the directive with an IP address or a host name. For details, see the description of these directives in "Access control - Set up access control for the server" on page 435.

Part 6. Programming

Chapter 18. Writing Common Gateway Interface programs

Overview of the CGI 315
CGI and dynamic documents 316
Uses for CGI 316
FastCGI support 317
Overview. 317
Updates to the Web server configuration file for FastCGI 317
Updates to the FastCGI configuration file . . . 317
DefaultUser directive 318
Local directive and subdirectives 318
External directive and subdirectives 319
Installing Version 2.2 of the FastCGI Developer's Kit 320
Step 1. Download the FastCGI Developer's Kit to your workstation 320
Step 2. Transfer the devkit_2_2_0_tar.tar file to z/OS 320
Step 3. Uncompress and reload the devkit_2_2_0_tar.gz file 320
Step 4. Rebuild the FastCGI Developer's Kit on z/OS 321
Hints and tips 322
Installing Version 2.1 of the FastCGI Developer's Kit 322
Step 1. The FastCGI Developer's Kit and the gzip utility 322
Step 2. Transfer the fcgi-devkit-2_1_tar.gz file to z/OS 323
Step 3. Uncompress and reload the fcgi-devkit-2_1_tar.gz file 323
Step 4. Rebuild the FastCGI Developer's Kit on z/OS 323
Hints and tips 325
Forms and data processing 325
Creating HTML documents that reference CGI programs. 326
Example 327
Sending information to the server 328
Processing the information 328
Parsing 328
Data manipulation 329
Response generation 329
Returning output 331
Protecting your programs 331
Environment variables 331
Processing standard search (ISINDEX) documents 331
Passing SSL environment variables to a CGI program 332
Creating CGI source 332
Parsing Routines 332
String Compare Reminders. 332
Running CGI programs written in Java™ 333
Response generation 334
CGI Examples 335
CGI C program. 335
Example (cgixmp) 335
CGI REXX program 340
Example 340
CGI shell script. 342
Example 342

Overview of the CGI

CGI is a standard, supported by almost all Web servers, that defines how information is exchanged between a Web server and an external program (CGI program). CGI programs are stored in the cgi-bin subdirectory.

CGI programs can be written in any language supported by the operating system on which the server is run. The language can be a programming language, like C++, or it can be a scripting language, like Perl or REXX. The HTTP Server also supports CGI programs written in the Java™ language. Programs written in programming languages need to be compiled, and typically run faster than uncompiled programs. On the other hand, those written in scripting languages tend to be easier to write, maintain, and debug.

The functions and tasks that CGI programs can perform range from the simple to the very advanced. In general, those that perform the simple tasks are called **CGI scripts** (because they are not compiled). Those that perform more complex tasks are often called **gateway programs**. In this chapter, we refer to both types as **CGI programs**.

There are many uses for CGI programs. Basically, they are designed to handle **dynamic information**. Dynamic in this context refers to temporary information that is created for a one-time use and not stored anywhere on the Web. This information may be a document, an e-mail message, or the results of a conversion program.

CGI and dynamic documents

There are many types of files that exist on the Web. Primarily they fall into one of the following categories:

- Images
- Multimedia
- Programs
- HTML documents

Servers break HTML documents into two distinct types:

- Static
- Dynamic

Static documents exist in source form on the Web server. **Dynamic documents** are created as temporary documents to satisfy a specific, individual request.

Consider the process of "serving" these two types of documents. Responding to requests for static documents is fairly simple. For example, Jill User accesses the Acme Web server to get information on the Pro-Expert gas grill. She clicks on Products, then on Grills, and finally on Pro-Expert. Each time Jill clicks on a link, the Web browser uses the URL attached to the link to request a specific document from the Web server and the server responds by sending a copy of the document to Jill's browser.

But, what if Jill then decides she wants to search through the information on the Acme Web server for all documents that contain information on Acme grills, such as news articles, press releases, price listings, and service agreements? This is a more difficult request to process. This is not a request for an existing document. Instead, it is a request for a dynamically generated list of documents that meet certain criteria. This is where CGI comes in.

Uses for CGI

You can use a CGI program to parse the request, search through the documents on your Web server, and create a list with hypertext links to each of the documents that contain the specified word or string.

HTML allows you to access resources on the Internet using many protocols by specifying the protocol in the URL. One such protocol is ***mailto***. If your HTML document includes a mailto link followed by an e-mail address, the link will result in a generic mail form.

What if you wanted your customers to provide specific information, such as how often they use the Web or how they heard about your company? Rather than using the generic mail form, you can create a form that asks these questions and more. You can then use a CGI program to interpret the information, include it in an e-mail message, and send it to the appropriate person.

CGI programs are not limited to processing search requests and e-mail. They can be used for a wide variety of purposes. Basically, anytime you want to take input from the browser and generate a response, you can use a CGI program. For example, many people are interested in how often people have visited their home

page. A common way to track this is with a CGI program that counts the number of requests for this home page and displays the new total each time someone links to it.

Note: Be aware that using CGI to serve documents can significantly impact your server's performance. Serving a document using CGI can take as much as 10 times more resources than serving a static document.

FastCGI support

Overview

FastCGI is a way to combine the advantages of normal CGI programming with some of the performance benefits you get by using the GWAPI interface.

FastCGI is written by Open Market, Inc. and is an extension to normal Web server processing. FastCGI allows you to start applications in independent address spaces and to pass requests for these applications from the Web server to these processes. The communication is handled by the TCP/IP sockets interface or by a UNIX domain socket bind path in the HFS file system.

By default, FastCGI support in the Web server is disabled. To enable support, you need to define directives in the following configuration files:

- Web server configuration file: httpd.conf
- FastCGI configuration file: lgw_fcgi.conf

For instructions, see "Updates to the Web server configuration file for FastCGI" and "Updates to the FastCGI configuration file".

You must also download and install the Open Market toolkit. The FastCGI Developer's Kit, which includes the toolkit, sample applications, and information on FastCGI, can be found on the Open Market Web site at URL:

```
http://www.fastcgi.com/
```

For instructions on installing the Developer's Kit on z/OS, see "Installing Version 2.1 of the FastCGI Developer's Kit" on page 322.

Updates to the Web server configuration file for FastCGI

Directives, their paths, and libraries that are essential for the operation of FastCGI support are:

```
ServerInit  /usr/lpp/internet/bin/libfcgi.so:FCGIInit /etc/lgw_fcgi.conf
Service     /fcgi-bin/* /usr/lpp/internet/bin/libfcgi.so:FCGIDispatcher*
ServerTerm  /usr/lpp/internet/bin/libfcgi.so:FCGIStop
```

Notes:

1. The ServerInit directive specifies the location of the FastCGI configuration file, lgw_fcgi.conf.
2. The Service directive indicates the path of the FastCGI object library.
3. The ServerTerm directive stops the FastCGI process initialization if the Web server is restarted.

Updates to the FastCGI configuration file

All FastCGI applications have to be registered in the lgw_fcgi.conf file. An example of this file is in the /usr/lpp/internet/samples/config directory. All these

applications are started during server initialization so you must stop and restart the Web server to use updates made after initialization.

In general, FastCGI supports two types of applications:

- **Local**

  The FastCGI task runs on the same system as the Web server. Communication is handled by using a dedicated TCP/IP socket or a BindPath in a character special HFS file.
- **External**

  The FastCGI task runs on a remote system. Communication can only be handled using a dedicated TCP/IP socket. Note that you need a Web server with FastCGI server support to control the startup and closedown of the application address spaces on the remote system.

You use Local and External directives to identify FastCGI applications in the lgw_fcgi.conf file.

Note: You can use more than one Arg or Environ subdirective to pass multiple values. For all other subdirectives, the Web server uses only the last subdirective in the configuration file.

DefaultUser directive

DefaultUser *user name*

Default user identity that the FastCGI will run under if you do not specify a user ID on the User subdirective of the Local directive.

Note: If you use the Bindpath option, the default user identity will be overriden with the propogated user ID.

Local directive and subdirectives

The Local directive indicates that the FastCGI application is on the same workstation as the Web server and that the Web server will start and control the application. Subdirectives are:

Exec *fully-specified path*

Required. Name and path of the FastCGI application.

Role *Responder* | *Authorizer*

Required. Type of FastCGI application:
- Responder applications perform like a normal CGI program.
- Authorizer applications perform user authentication.

URL *server URL*

Required. The relative uniform resource location of the FastCGI application.

BindPath | **Port** *Unix domain socket bind path* | *Port number*

Required. The path name of the file or the port number which will be used to communicate between the Web server and the FastCGI application.

If you use the Port subdirective, the request will run under the user ID that was specified on the User subdirective of the Local directive. If you specify a user ID on neither the User subdirective nor the DefaultUser directive, the request runs under the user ID of the server address space. If you do not specify a user ID on the User subdirective, but you do specify a user ID on DefaultUser, the request runs under the default user ID instead of the ID of the server address space.

If you use the BindPath subdirective, the authority of the thread that is handling the request will be propagated to the FastCGI address space. The request in the FastCGI will run under this propagated authority. The UserID subdirective in a Protection directive or a DefProt directive determines the ID and authority of the thread. However, if you do not define the ID on this UserID subdirective, the default user ID defined on the UserID directive determines the user ID and authority of the thread.

Note: The user ID specified on either the DefaultUser directive or the User subdirective of the Local directive determines the owner of the FastCGI address space. This user ID is not necessarily the same as the user ID under which the FastCGI request is run.

Note: The httpd.conf file contains the definitions for the UserID directive and the UserID subdirective. The FastCGI configuration file of lgw_fcgi.conf contains the definition for the User subdirective.

Note: For Scalable Mode you must specify a BindPath (UNIX domain socket bind path). Port (port number) will not work in Scalable Mode.

NumProcesses *number*
Optional. The number of processes the Web server will create for the FastCGI application. The default is one.

NPH-Script*flag*
Optional. Specifies that the FastCGI application will produce nonparsed header output.

Arg *command line argument*
Optional. Allows the user to enter additional command line arguments to start the FastCGI application.

Environ *environment_variable=value*
Optional. Allows the user to set environment variables to pass to the FastCGI application at initialization.

User *userID*
Optional. Identifies the userID of the FastCGI application.

The DefaultUser directive can be used to identify the userID of the FastCGI application if a User directive is not specified. The **User** subdirective overrides the DefaultUser directive

For more information on the options available for the Local directive, see the comments in the FastCGI configuration file: lgw_fcgi.conf

External directive and subdirectives

The External directive identifies a FastCGI application that resides on a remote workstation. Subdirectives are:

URL *server URL*
Required. The relative uniform resource location of the FastCGI application.

Host *host name or IP address*
Required. The host name or IP address of the workstation the Web server should connect to.

Port *port number*
Required. The port number which will be used to communicate between the Web server and the FastCGI application.

Role *Responder* | *Authorizer*
Required. Type of FastCGI application:
- Responder applications perform like a normal CGI program.
- Authorizer applications perform user authentication.

NPH-Script *flag*
Optional. Indicates that the FastCGI application produces nonparsed header output.

For more information on the options available for the External directive, see the comments in the FastCGI configuration file: lgw_fcgi.conf

Installing Version 2.2 of the FastCGI Developer's Kit

The current implementation of the FastCGI Developer's Kit from Open Market, Inc., supports programs written in C/C++, Perl, and Java™.

Note: The Perl and Java FastCGI implementations have to be modified for use on z/OS. At the time of this writing, there were no ported versions available.

To install the C version of the FastCGI Developer's Kit:

Step 1. Download the FastCGI Developer's Kit to your workstation

Download the Developer's Kit from the Open Market Web site at URL:

```
http://www.fastcgi.com/
```

The Developer's Kit is distributed as a UNIX compressed gzip file and requires the gzip utility to uncompress it into a tar file. To download the gzip utility, go to the z/OS UNIX Tools and Toys Web page at URL:

```
http://www.ibm.com/s390/oe/bpxa1toy.html
```

Click **Ported Tools** to access the download page.

To install the gzip utility, use the installation instructions on the Web site. Make sure your REGION size is large enough to run the make command. A size of 48000 in the TSO logon screen was used when testing this procedure. To unpack the gzip utility, issue the following command:

```
pax -o from=ISO8859-1,to=IBM-1047 -rf gzip-1_2_4.tar
```

Hints and tips: If you continue to use the same shell to build the FastCGI library after creating the gzip utility, the Makefile command will fail. You will see an error message regarding a file syntax error. This is because the utility uses the c89 compiler command instead of cc. To eliminate this problem, either open another shell or export CC=cc and continue.

Step 2. Transfer the devkit_2_2_0_tar.tar file to z/OS

To copy the devkit_2_2_0_tar.tar file from your workstation to the UNIX HFS file system on z/OS, you can use either binary FTP or PC file transfer. During the file transfer rename the file to devkit_2_2_0_tar.gz.

Step 3. Uncompress and reload the devkit_2_2_0_tar.gz file

Run the gzip utility to unpack the devkit_2_2_0_tar.gz file by issuing the following command:

```
gzip -d devkit_2_2_0_tar.gz
```

The tar file produced (devkit_2_2_0_tar) contains text files in ASCII format. In UNIX System Services, EBCDIC is normally used. You must untar the file and translate the resulting files from ASCII to EBCDIC.You can do this using either the UNIX shell script ext.sh **or** the pax command.

You can get a copy of ext.sh from the z/OS UNIX Tools and Toys Web page at URL: :

```
http://www.ibm.com/s390/oe/bpxa1toy.html
```

To execute the ext.sh shell script, issue:

```
ext.sh -v devkit_2_2_0_tar
```

Note: If the ext.sh script is executed and the following messages are displayed:

```
[: ext.sh 37: FSUM6807 expression syntax error
No binary files found]
```

the ext.sh script needs to be changed. Edit the ext.sh script:

1. Insert after line 26:

   ```
   binaryFiles=NONE
   ```

2. Change line 29 from:

   ```
   if [ $binaryFiles ]; then
   ```

   to

   ```
   if [ "$binaryFiles" != "NONE" ]; then
   ```

3. Re-execute the ext.sh script.

To execute the pax command instead, issue:

```
pax -o from=ISO8859-1,to=IBM-1047 -rf devkit_2_2_0_tar
```

Step 4. Rebuild the FastCGI Developer's Kit on z/OS

1. Execute the following two commands:
   a. Change to the devkit_2.2.0 directory:

      ```
      cd <root>/devkit_2.2.0
      ```

   b. Execute the configure program:

      ```
      ./configure
      ```

2. Change files in the libfcgi directory:
   a. change to the libfcgi directory

      ```
      cd <root>/devkit_2.2.0/libfcgi
      ```

   b. edit os_unix.c by changing line 37

      ```
      from: #include <netinet/tcp.h>
      to: #include <xti.h>
      ```

   c. edit Makefile by changing line 29

      ```
      from: $(RANLIB) libfcgi.${L}
      to: # $(RANLIB) libfcgi.${L}
      ```

3. Change the examples Makefile
   a. Change to the examples directory

      ```
      cd <root>/devkit_2.2.0/examples
      ```

   b. Change line 16

      ```
      from: CFLAGS = -g -I$(INCLUDEDIR)
      to:   CFLAGS = -g -I$(INCLUDEDIR) -DNDEBUG
      ```

   c. Change line 32

```
from:
$(CC) $(CFLAGS) -D_REENTRANT threaded.c
-o threaded.fcg $(LIBFCGI) $(LIBS) -lpthread
to:
$(CC) $(CFLAGS) -D_REENTRANT threaded.c
-o threaded.fcg $(LIBFCGI) $(LIBS)
```

Note: Line 32 in Makefile is shown here on multiple lines.

d. Change line 70

```
from:
rm -f *.${L} *.${O} core.* errs *~ \#*
TAGS *.E a.out $(TARGETS)
to:
rm -f *.${L} *.${O} core.* errs *~ "\#*"
TAGS *.E a.out $(TARGETS)
```

Note: Line 70 in Makefile is shown here on multiple lines.

4. Execute the following commands:
   a. Change to the devkit_2.2.0 directory

      ```
      cd <root>/devkit_2.2.0
      ```

   b. `export _CC_CCMODE=1`
   c. Execute the make program

      ```
      make
      ```

You now have a full copy of the Developer's Kit on your z/OS system.

Note: To redo the make for the entire tool-kit

1. Change to the devkit_2.2.0 directory:

   ```
   cd <root>/devkit_2.2.0
   ```

2. Execute the cleanup make

   ```
   make clean
   ```

3. Execute the build make

   ```
   make
   ```

Hints and tips

If the compile fails at any stage, you must run the make clean command, fix the error, and run the make command again.

Installing Version 2.1 of the FastCGI Developer's Kit

Version 2.1 of the FastCGI Developer's Kit from Open Market, Inc., supports programs written in C/C++, Perl, and Java™.

Note: The Perl and Java FastCGI implementations have to be modified for use on z/OS. At the time of this writing, there were no ported versions available.

To install the C version of the FastCGI Developer's Kit:

Step 1. The FastCGI Developer's Kit and the gzip utility

Version 2.1 is a back level of the FastCGI Developer's Kit. However, if you already have a copy of version 2.1, you can install and use it. These steps assume that the Version 2.1 FastCGI Developer's Kit is stored on your workstation.

The Developer's Kit is distributed as a UNIX compressed gzip file and requires the gzip utility to uncompress it into a tar file. To download the gzip utility, go to the z/OS UNIX Tools and Toys Web page at URL:

```
http://www.ibm.com/s390/oe/bpxa1toy.html
```

Click **Ported Tools** to access the download page.

To install the gzip utility, use the installation instructions on the Web site. Make sure your REGION size is large enough to run the make command. A size of 48000 in the TSO logon screen was used when testing this procedure. To unpack the gzip utility, issue the following command:

```
pax -o from=ISO8859-1,to=IBM-1047 -rf gzip-1_2_4.tar
```

Hints and tips: If you continue to use the same shell to build the FastCGI library after creating the gzip utility, the Makefile command will fail. You will see an error message regarding a file syntax error. This is because the utility uses the c89 compiler command instead of cc. To eliminate this problem, either open another shell or export CC=cc and continue.

Step 2. Transfer the fcgi-devkit-2_1_tar.gz file to z/OS

To copy the file from your workstation to the UNIX HFS file system on z/OS, you can use either binary FTP or PC file transfer.

Step 3. Uncompress and reload the fcgi-devkit-2_1_tar.gz file

Run the gzip utility to unpack the fcgi-devkit-2_1_tar.gz file by issuing the following command:

```
gzip -d fcgi-devkit-2_1_tar.gz
```

The tar file produced (fcgi-devkit-2_1_tar) contains binary files and text files in ASCII format. In UNIX System Services, EBCDIC is normally used. Therefore, for some of the files, you use the pax command with a translation option, and a binary transfer for others.

You can use the UNIX shell script (ext.sh) to control the untar process. This script uses the translation option only on text files. You can get a copy of this script on the z/OS UNIX Web site at URL:

```
http://www.ibm.com/s390/oe/bpxa1toy.html
```

The following command creates subdirectory fcgi-devkit-2.1 in the current working directory for the Developer's Kit:

```
pax -o from=ISO8859-1,to=IBM-1047 -rf fcgi-devkit-2_1_tar
```

Step 4. Rebuild the FastCGI Developer's Kit on z/OS

1. Execute the following two commands:

   ```
   cd fcgi-devkit-2.1
   ./configure
   ```

2. Edit examples/sample-store.c

   Change lines 504 and 524

   ```
   from: cart = Tc1_GetHashValue(cartEntry);
   to: cart = (CartObj *)Tc1_GetHashValue(cartEntry);
   ```

   Change lines 532, 744, 758, 803, and 825

   ```
   from: CartObj *cart = Tc1_GetHashValue(cartEntry);
   to: CartObj *cart = (CartObj *)Tc1_GetHashValue(cartEntry);
   ```

3. Edit libfcgi/os_unix.c to change line 54

   ```
   from:#include <netinet/tcp.h>
   to:#include <xti.h>
   ```

4. Edit libfcgi/fcgi_stdio.c

Change line 51

```
from:#ifdef _WIN32
to:/*#ifdef _WIN32*/
```

Change line 53

```
from:#else
to:/*#else
```

Change line 55

```
from:#endif
to:#endif*/
```

5. Edit the libfcgi/Makefile to change line 29

```
from: $(RANLIB) libfcgi.${L}
to:# $(RANLIB) libfcgi.${L}
```

6. Edit the include/fcgi_stdio.h file to change all occurrences of X'0D' X'40'. If you are editing using OEDIT, you can use the command `c all X'0D' X'40'`.
7. Edit cgi-fcgi/Makefile to change line 30

```
from:$(CC) $(CFLAGS) cgi-fcgi.${O} -o cgi-fcgi $(LIBFCGI) $(LIBS)
to:$(CC) $(CFLAGS) -o cgi-fcgi cgi-fcgi.${O} $(LIBFCGI) $(LIBS)
```

8. Edit examples/Makefile

Change line 16

```
from: CFLAGS = -g -I$(INCLUDEDIR)
to: CFLAGS = -g -I$(INCLUDEDIR) -DNDEBUG
```

Change line 35

```
from: $(CC) $(CFLAGS) tiny-cgi.${O} -o tiny-cgi.cgi
to:  $(CC) $(CFLAGS) -o tiny-cgi tiny-cgi.${O}
```

Change line 38

```
from: $(CC) $(CFLAGS) tiny-fcgi.${O} -o tiny-fcgi $(LIBFCGI) $(LIBS)
to:  $(CC) $(CFLAGS) -o tiny-fcgi tiny-fcgi.${O} $(LIBFCGI) $(LIBS)
```

Change line 41

```
from: $(CC) $(CFLAGS) tiny-fcgi2.${O} -o tiny-fcgi2 $(LIBFCGI) $(LIBS)
to:  $(CC) $(CFLAGS) -o tiny-fcgi2 tiny-fcgi2.${O} $(LIBFCGI) $(LIBS)
```

Change line 44

```
from: $(CC) $(CFLAGS) tiny-authorizer.${O} -o tiny-authorizer $(LIBFCGI) $(LIBS)
to:  $(CC) $(CFLAGS) -o tiny-authorizer tiny-authorizer.${O} $(LIBFCGI) $(LIBS)
```

Change line 47

```
from: $(CC) $(CFLAGS) echo.${O} -o echo $(LIBFCGI) $(LIBS)
to:  $(CC) $(CFLAGS) -o echo echo.${O} $(LIBFCGI) $(LIBS)
```

Change line 50

```
from: $(CC) $(CFLAGS) echo2.${O} -o echo2 $(LIBFCGI) $(LIBS)
to:  $(CC) $(CFLAGS) -o echo2 echo2.${O} $(LIBFCGI) $(LIBS)
```

Change line 53

```
from:
$(CC) $(CFLAGS) sample-store.${O} tclHash.{O} -o sample-store $(LIBFCGI) $(LIBS)
to:
$(CC) $(CFLAGS) -o sample-store sample-store.${O} tclHash.{O} $(LIBFCGI) $(LIBS)
```

Change line 56

```
from: $(CC) $(CFLAGS) log-dump.${O} -o log-dump $(LIBFCGI) $(LIBS)
to:  $(CC) $(CFLAGS) -o log-dump log-dump.${O} $(LIBFCGI) $(LIBS)
```

9. Execute the following two commands:

```
cd fcgi-devkit-2.1
make
```

You now have a full copy of the Developer's Kit on your z/OS system.

Hints and tips

If the compile fails at any stage, you must run the make clean command, fix the error, and run the make command again.

Forms and data processing

The CGI process involves three players: the Web browser, the Web server, and the CGI program. The CGI Sample Test Case shown here is an HTML form that uses a CGI program to process the input. Let's assume that the Web browser has already requested and obtained this document.

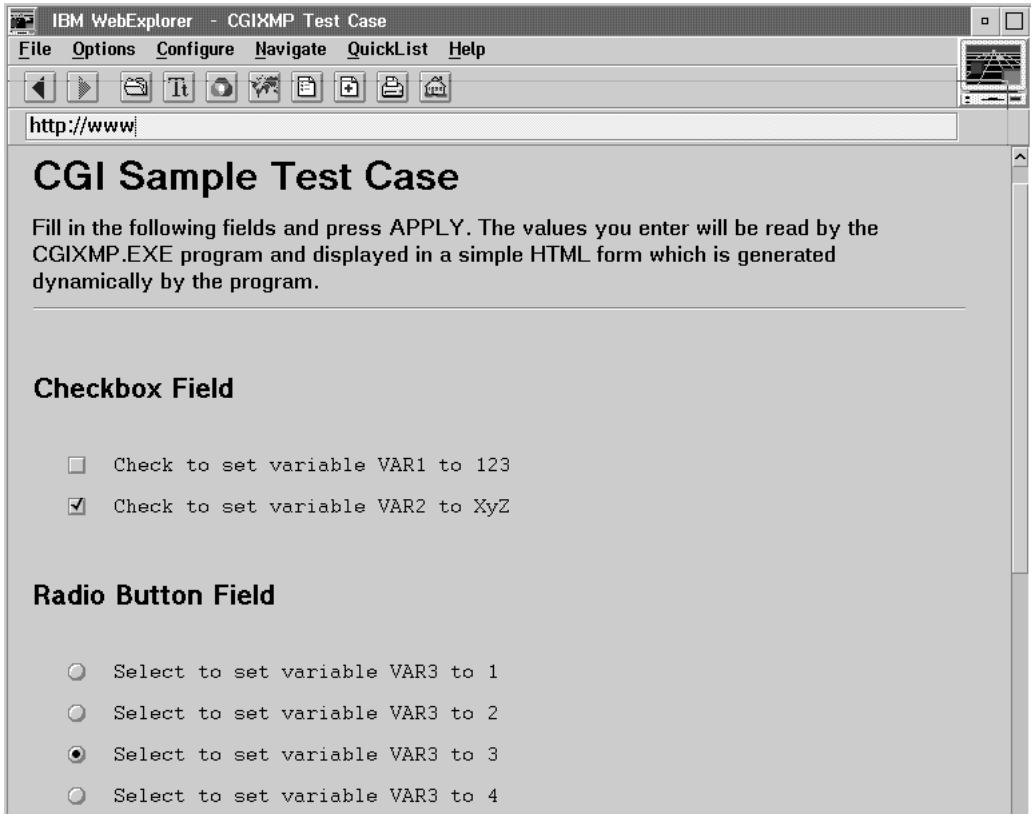


Figure 2. Sample Form, Page 1

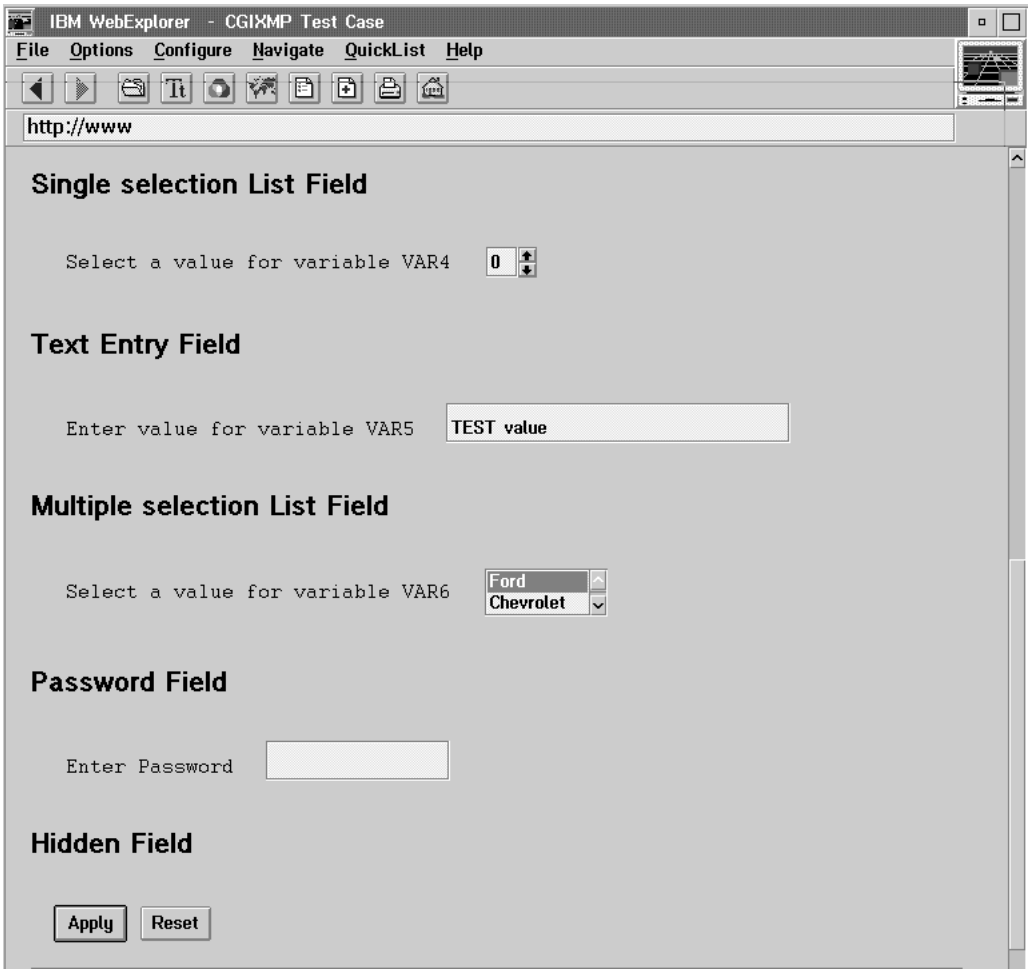


Figure 3. Sample Form, Page 2

The following sequence of events shows the roles played by the Web browser, Web server, and CGI program in processing this form:

1. The user fills out the form (clicks buttons or enters information in appropriate fields) and then clicks the Submit button.
2. The Web browser sends the data to the Web server in an encoded format. In our example, this data consists of responses the user entered on this form.
3. Upon receiving the data, the Web server converts the data to a format compliant with the CGI specification for input and sends it to the CGI program.
4. The CGI program decodes the data and processes it per its instruction.
5. The CGI program sends the response back to the Web server in a form that is compliant with the CGI specification for output.
6. The Web server interprets the response and forwards it to the Web browser.

Creating HTML documents that reference CGI programs

CGI programs are referenced from within HTML documents. When you design the layout of your HTML document, keep in mind how a CGI program might affect the look of your document. Developing the CGI program along with the HTML document can help you avoid many design mistakes.

The following HTML source produces our CGI Sample Test Case shown in Figure 2 on page 325. It shows the use of the HTML <form> tag.This tag defines the form within the document and specifies both the HTTP method by which the data will be sent to the server (POST) and the URI of the CGI program that will process the data (/cgi-bin/cgixmp). The following HTML source also shows how to code various types of input fields that you can include in your HTML forms.

Example

```
<HTML>
<HEAD>
<TITLE>CGIXMP Test Case</TITLE>
</HEAD>
<BODY>
<H1>CGI Sample Test Case</H1>
Fill in the following fields and press Submit.
The values you enter will
be read by the CGIXMP program and displayed in a simple HTML
form which is generated dynamically by the program.
<P>
<HR>
<form method=POST action="/cgi-bin/cgixmp">
<P>
<H3>Checkbox Field</H3>
<P>
<PRE>
<input type="checkbox" name="var1" value="123">
Check to set variable VAR1 to 123<BR>
<input type="checkbox" name="var2" value="XyZ" checked>
Check to set variable VAR2 to XyZ<BR>
</PRE>
<P>
<H3>Radio Button Field</H3>
<P>
<PRE>
<input type="radio" name="var3" value="1">
Select to set variable VAR3 to 1<BR>
<input type="radio" name="var3" value="2">
Select to set variable VAR3 to 2<BR>
<input type="radio" name="var3" value="3" checked>
Select to set variable VAR3 to 3<BR>
<input type="radio" name="var3" value="4">
Select to set variable VAR3 to 4<BR>
</PRE>
<P><H3>Single selection List Field</H3>
<P>
<PRE>
Select a value for variable VAR4   <select size=1 name="var4">
<option>0<option>1<option>2<option>3
<option>4<option>5</select>
</PRE>
<P>
<H3>Text Entry Field</H3>
<P>
<PRE>
Enter value for variable VAR5 <input type="text" name="var5"
size=20 maxlength=256 value="TEST value">
</PRE>
<P>
<H3>Multiple selection List Field</H3>
<P>
<PRE>
Select a value for variable VAR6
<select multiple size=2 name="var6">
<option>Ford<option>Chevrolet<option>Chrysler<option>
Ferrari<option>Porsche
</select>
```

```
</PRE>
<P>
<H3>Password Field</H3>
<P>
<PRE>
Enter Password
<input type="password" name="pword" size=10 maxlength=10>
</PRE>
<P>
<H3>Hidden Field</H3>
<P>
<input type="hidden" name="hidden" value="Text not shown on form...">
<P>
<PRE>
<input type="submit" name="pushbutton" value="Submit">
<input type="reset" name="pushbutton" value="Reset">
<HR>
</PRE>
</BODY>
</HTML>
```

Sending information to the server

When a user fills out an HTML form or enters a phrase in a search field and clicks on the Submit button, the Web browser sends the request to the server in a format described as ***URL-encoded***.

In URL-encoded information:

- The URL starts with the URL of the processing program.
- Attached data, such as name=value information from a form, is appended to the URL and preceded by a question mark.
- Fields are separated by an ampersand (&).
- Space are represented by a plus sign (+).
- Special characters, such as a period or slash, are represented by a percent sign (%) followed by the ASCII hexadecimal equivalent of the symbol.
- Multiple values for a field, such as check boxes, are sent as a string separated by an ampersand (&).

Upon receiving the data, the Web server converts it to a format compliant with the CGI input specification and passes it to the CGI program specified in the HTML document.

Processing the information

Most CGI programs process the data they receive in the following three stages:

- Parsing
- Data manipulation
- Response generation

Parsing

Parsing is the first stage of a CGI program. In this stage, the program takes the data in one or more of the possible formats (environment variables, command-line arguments, or standard input devices), breaks it into components, and decodes the information in the components.

For example, the following could be received using the environment variable QUERY_STRING:

```
NAME=Eugene+T%2E+Fox&ADDR=etfox%40ibm.net&INTEREST=RCO
```

Parsing breaks the fields at the ampersands and decodes the ASCII hexadecimal characters. The results look like this:

```
NAME=Eugene T. Fox
ADDR=etfox@ibm.net
INTEREST=RCO
```

You can use the cgiparse command to automatically parse query strings, read and write CONTENT_LENGTH characters, and count the number of form fields submitted. For a complete description of the cgiparse command, see "cgiparse command" on page 384. Also, see the sample code in "CGI Examples" on page 335.

Data manipulation

Data manipulation is the second stage of a CGI program. In this stage, the CGI program takes the parsed data and performs the appropriate action. For example, a CGI program designed to process an application form might:

1. Take the input from the parsing stage
2. Convert abbreviations into more meaningful information
3. Plug the information into an e-mail template
4. Call the sendmail program
5. Send the filled-in template to a specified e-mail address

Response generation

Response generation is the final stage of a CGI program. In this stage, the CGI program formulates its response to the Web server, which forwards it to the Web browser. The response begins with **MIME** (Multipurpose Internet Mail Extensions) headers that give meta-information about the returned document. Depending on the type of response, the headers may vary, but the response must contain at least one MIME header (Content-Type) followed by a blank line. The blank line separates the headers from the content of the response.

Static documents: If the response is a static document, the program returns the URL of the document using the HTTP Location header, followed by a blank line. For example:

```
Location: http://www.acme.com/products.html
```

Upon receiving this information from the CGI program, the Web server will retrieve the specified document and send a copy of it to the Web browser. Using relative URLs instead of full URLs can improve the response time for clients.

Dynamic documents: If the response is a dynamic document, such as a list of hypertext links to documents that meet specified criteria, the program should indicate that the response is an HTML document, using the Content-Type header followed by a blank line, and then include links to the documents in HTML format. With a search, the links might be the URLs of all the documents that met the search criteria.

If the response is an HTML file, the program should indicate that the response is an HTML file, using the Content-Type header followed by a blank line, and then the body of the document.

With a request that results in e-mail, the response might be a message confirming that the e-mail was sent.

Using nph: If you do not want the Web server to interpret the response, but just to forward it to the Web browser, the name of your CGI program must begin with

nph- (no-parse header). A no-parse header program creates output that is a complete HTTP response, including the HTTP return code and status information. It requires no further action, such as interpretation or modification, on the part of the server.

Use the meta refresh HTML tag rather than nph if you want to periodically refresh the page because nph actually degrades performance. Possible reasons for using nph are:

- When returning output over a slow period of time, such as displaying unbuffered results of a slow operation in real time
- if you are implementing a server push of a multipart object such as a dynamically generated animated gif

Note: it is the responsibility of the **nph-** script writer to generate the entire HTTP response, including the HTTP status line.

HTML output: The response from our CGI Sample Test Case produces the following HTML file that shows the Content-Type header:

```
Content-Type: text/html

<HTML>
<HEAD>
<TITLE>Test HTML Page</TITLE>
</HEAD>
<BODY>
<H1>Variable Information</H1>
<HR>
<P>
<PRE>Variable "var1" = 123</PRE>
<P><PRE>Variable "var2" = XyZ</PRE>
<P><PRE>Variable "var3" = 3</PRE>
<P<PRE>Variable "var4" = 0</PRE>
<P><PRE>Variable "var5" = TEST value</PRE>
<P><PRE>Variable "var6" = Ford</PRE>
<P><PRE>Variable "pword" = </PRE>
<P><PRE>Variable "hidden" = Text not shown on form...
</PRE>
<P><PRE>Variable "pushbutton" = Submit</PRE>
<P><PRE><P>
<HR>
</BODY>
</HTML>
```

Producing headers with cgiutils: You can use the cgiutils command in your CGI program to produce full or partial sets of MIME headers. You can use optional flags with this command to keep the date from appearing or to display version information, expiration information, and more.

This example shows some of the MIME headers you can produce:

```
HTTP/1.0 200 OK
MIME-Version:1.0
Date: Monday, 25 Oct 95 13:14:15 GMT
Content-Type:text/html
Document-Name:/tcpip/form.html
Expires: Wednesday, 25 Oct 95 13:14:15 GMT
```

For more information on using the cgiutils command and its flags to create MIME headers, see “cgiutils command” on page 387.

Returning output

When the CGI program is finished, it passes the resulting response to the Web server using standard output (stdout). The Web server interprets the response and sends it to the Web browser.

If the CGI program encounters errors, it may write error information to standard error (stderr). The HTTP Server redirects stderr to the cgi_error log.

For more information on the ErrorLog directive, see "ErrorLog - Name the file where you want to log internal server errors" on page 503.

Protecting your programs

Storing your programs in the cgi-bin subdirectory provides some level of protection because you can ensure that typical users do not have write access to it. In most cases, this is sufficient. However, there are some devious users out there who can figure out how to use your CGI program to "break into" your server.

You can guard against this by understanding the limitations of your code. For instance, if a Perl program encounters an escape character as input, it will abort and dump the user out to the machine's root directory. If you do not compensate for this, a user could send escape characters in the URL-encoded data as input and gain access to your server. The easiest way to guard against this is by using compiled programs instead of scripts. You can use protection setups and ACL files to protect your programs as well.

For information on access control settings for the Web server, see "Access control - Set up access control for the server" on page 435.

Environment variables

Before writing your CGI program, you need to understand the format in which the server will pass the data. The server receives the URL-encoded information and, depending on the type of request, passes the information to the CGI program using environment variables, command line arguments, or standard input.

In general, the HTML document defines the environment variables that specify how information is passed. For all requests, regardless of type, certain information is passed using the environment variables.

For a description of environment variables you can use in CGI programs, see Appendix D, "Environment variables" on page 583.

Processing standard search (ISINDEX) documents

ISINDEX is an HTML tag that identifies the document as a standard search document and causes the browser to automatically generate an entry field. When information is sent from an ISINDEX document, the server takes the appended data (the information following the ?), breaks it at the pluses (+), and sends the data to the CGI program as command line arguments (argv). For example:

```
<ISINDEX>
```

Note: It is possible to write CGI programs that display all environment variables. At times these variables may include sensitive data such as user IDs and

passwords for various products. So you must be careful about displaying environment variables in your CGI programs and you must be careful about who has access to them.

Passing SSL environment variables to a CGI program

For a list of SSL-related environment variables you can use in CGI programs, see Appendix D, “Environment variables” on page 583.

Creating CGI source

CGI programs can be written in any language supported by the operating system on which the server is running. The examples in this chapter show CGI programs written in C, REXX, and shell script. To review these examples, see “CGI Examples” on page 335.

The z/OS operating system has a requirement for translating characters from EBCDIC to ASCII encoding. For an example using this translation, see “CGI C program” on page 335.

To access coding samples, go to the Web site Library page at URL:

`http://www.ibm.com/software/webservers/httpservers/doc53.html`

Parsing Routines

On z/OS, an additional complication with CGI programs is that most clients run on ASCII systems, while z/OS uses EBCDIC encoding of characters.

Note that:

- for most characters sent from a browser to the HTTP Server, the web server will translate the character from ASCII to EBCDIC.
- for special characters, the browser “escapes” the character prior to sending it to the web server. For example, the ? character is changed to %3F, x’3F’ being the hexadecimal representation in ASCII of ?. The web server receives the character string %3F and converts **the string** to EBCDIC. Your CGI program is then responsible for translating the string back to a single character. The supplied sample CGI program has a routine TranslateEscapes, which converts the escaped characters back to **ASCII**. x’3F’ in EBCDIC is not a valid character. The hexadecimal representation of ? in EBCDIC is x’6F’.

Most CGI programs z/OS Web servers do not take this into account, and therefore will not process special characters entered in an HTML form correctly. The documentation provided with the HTTP Server does describe how to resolve this issue in your own CGI programs, but the point is important so warrants repeating.

String Compare Reminders

If you are trying to compare data passed by the user with some expected value; you will probably use the C function **strcmp()**. We found that the last value passed by the user can sometimes not match the expected value. It is good practice to use **strncmp()** and compare for a certain length, rather than using an unbounded compare. As an example, you should use code such as that shown following:

```
if (!strncmp(html, "yes", 3))
```

rather than

```
if (!strcmp(html, "yes"))
```

when you are comparing with a specific string value.

Another common problem we had when writing gateway programs was in processing logic based on the result of a compare, such as:

```
     if (!strncmp(html, "yes", 3))
     {
               ht = ON;
     }
else
     {
               ht = OFF;
     }
```

Note that if two strings compare equally, the result of the string compare is **0**, therefore if you want to perform logic such as that just shown, you must use the NOT form **(!strncmp)**.

Running CGI programs written in Java™

On the z/OS system, the HTTP Server depends on the settings of LIBPATH and CLASSPATH to successfully execute CGI programs written in the Java™ language. The server refers to the LIBPATH setting to dynamically load the z/OS Java run-time libraries and it consults the CLASSPATH setting to locate z/OS Java class libraries and the CGI programs. You can find these settings in /etc/httpd.envvars or its equivalent if you have overridden it with ENVAR("_CEE_ENVFILE=...").

You will need to modify these settings for your configuration. For example, if you have installed z/OS Java at /J1.1.8 and have placed your CGI program, myjavacgi.class, in the /usr/lpp/internet/server_root/cgi-bin directory, you will need to modify your environment variables file (/etc/httpd.envvars or its equivalent) as follows:

```
LIBPATH=/usr/lpp/internet/bin:/usr/lpp/internet/sbin:/J1.1.8/lib/mvs/native_threads
CLASSPATH=/usr/lpp/internet/server_root/cgi-bin:/J1.1.8/lib/classes.zip
```

Note: You must first compile your CGI programs written in Java with javac and then use the resulting class file, optionally including the .class suffix, as the name of the CGI program in your HTML file (for example, /cgi-bin/myjavacgi.class).You must also place the class file in a directory listed in the CLASSPATH environment variable. If you want to use a directory other than the ones currently in CLASSPATH, append this additional directory to the list in CLASSPATH, separated by a colon.

You must then update the server's configuration file with an Exec directive that will cause the Java CGI to be executed when a specific URL is received. For more information, see "Exec - Run a CGI program for matching requests" on page 537.

After setting up the server to run your Java CGI, attempt to run it from a browser. If the interface to Java fails, an error page may be returned to the browser with the following:

```
Error loading Java to run cginame (num)
```

cginame is the name of the Java CGI the server was asked to execute. *num* is a number that indicates the reason for the failure. *num* can be one of the following:

1 The requested class file did not have the .class extension.

2 The CLASSPATH environment variable was not set.

3 The LIBPATH environment variable was not set.

4 Error attempting to load the libjava.a run-time DLL. This can occur for several reasons:

- The correct path was not added to the LIBPATH environment variable for Java.
- The file libjava.a was not found in the path specified on the LIBPATH environment variable.
- Incorrect permission settings are set for file libjava.a or any directory it is contained in.

5 The entry point java_main could not be located.

6 The allocation of temporary storage failed.

7 The entry point iconv_init could not be located.

8 An error returned from the iconv_init() function.

9 The java_main was improperly returned to the interface routine.

An additional line of text is included in 4, 5, 7 and 8 that indicates the current values of errno and errno2.

Response generation

You also need to consider character translation in your CGI programs. Most browsers run on operating systems that use ASCII encoding of characters, while the z/OS system uses EBCDIC encoding. Therefore, on the z/OS system, you can also include a Content-Encoding header in your CGI response to specify whether or not the server should translate the content of the response before sending it back to the browser.

Recognized values for the Content-Encoding header are:

7bit All data is represented as short lines (less than 1000 characters) of 8859-1 ASCII data. Source code or plain text files usually fall into this category. Exceptions would be files containing line-drawing characters or accented characters.

8bit Data is represented as short lines, but may contain characters with the high bit set (for example, line-drawing characters or accented characters). PostScript files and text files from European sites usually fall into this category.

binary This encoding can be used for all data types. Data may contain not only non-ASCII characters, but also long (greater than 1000 characters) lines. Almost every file of type image/*, audio/*, and video/* falls into this category, as do binary data files of type application/*.

ebcdic Data is represented as lines of EBCDIC text (IBM-1047) and data is translated to ASCII text (ISO8859-1) when served. Note that codepages can be overridden by the DefaultFsCp and DefaultNetCp directives.

other Any *other* value of encoding receives the same treatment as binary.

If the Content-Encoding header contains the value `ebcdic`, the server will translate the content to ASCII characters. If the Content-Encoding header contains any other value, the server will not translate the content.

If the Content-Encoding header contains the value `7bit`, `8bit`, `binary` or `ebcdic`, the server will not forward the Content-Encoding header to the browser.

If the Content-Encoding header is missing, the server will use the information in the Content-Type header to determine if it should translate the content. In this case, if the Content-Type header specifies a value starting with `text/`, the server will translate the contents to ASCII characters. With any other value, it will not translate.

CGI Examples

This section includes the following examples:
- "CGI C program"
- "CGI REXX program" on page 340
- "CGI shell script" on page 342

CGI C program

The following example is a version of the C program, cgixmp, used in our CGI Sample Test Case shown in the example on page 327. It includes modifications to the TranslateEscapes routine which are required by the z/OS system for the conversion of escaped characters.

Most Web browsers run on ASCII based operating systems, while the z/OS system uses EBCDIC encoding of characters. When a browser sends data to the server running on the z/OS system, the server translates the characters from ASCII to EBCDIC encoding. But browsers "escape" special characters prior to sending them to the Web server and this can pose problems.

For example, when the browser escapes the ? character, it changes it to %3F before sending it (x'3F' is the hexadecimal representation in ASCII of ?). The z/OS HTTP Server receives the character string of %3F and has to convert the string to EBCDIC. The CGI program is then responsible for converting the EBCDIC string back to a single character. In this case, x'3F' is not a valid EBCDIC character; the valid representation of ? in EBCDIC is x'6F'. The conversion is applied in following example.

Example (cgixmp)

```
/* Includes */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define TRUE 1
/* Definition of structure used in linked list of variables */
typedef struct _argument
  {
  char *VariableName;
  char *Value;
  struct _argument *pNext;
  } PARMLIST, *PPARMLIST;

/* Function definitions */
int ErrMsg (char *msg);
PPARMLIST ReadArguments(int InputLength);
static void PlusesToSpaces(char *Str);
static int HexVal(char c);
static void TranslateEscapes(char *Str);
/*************************************************************************/
/*                                                                       */
/* Function       : main                                                 */
```

```
/*                                                                        */
/* Description    : This is a CGI program which takes the output of a form    */
/*                  submitted with a method of POST, and displays a list      */
/*                  of the variable names and values.                         */
/*                                                                        */
/**************************************************************************/
main(int argc, char *argv[])
  {
  char *requestMethod;
  char *contentLength;
  int argLength;
  PPARMLIST pParm = NULL;
  PPARMLIST pHead = NULL;
 /* This CGI program must be called with a method of POST */
  requestMethod = getenv("REQUEST_METHOD");

  if ((requestMethod == NULL) ||
      (strcmp(requestMethod, "POST")))
    {
    ErrMsg("REQUEST_METHOD environment variable not properly set to POST\n");
    }
  else
    {
    /* Get the length of the arguments passed in to this program */
    contentLength = getenv("CONTENT_LENGTH");

    if (contentLength == NULL)
      {
      ErrMsg("CONTENT_LENGTH environment variable not set\n");
      }
    else
      {
      /* Begin output of HTML to display results of CGI program */
      printf("Content-type: text/html\n\n");
      printf("<HMTL>\n");
      printf("<HEAD>\n");
      printf("<TITLE>Sample HTML Page</TITLE>\n");
      printf("</HEAD>\n");
      printf("<BODY>\n");
      printf("<H1>Variable Information</H1>\n");
      printf("<HR>\n");
      printf("<P>\n");
      /* Read the arguments passed in to this program and place them in */
      /* a singly linked list - one link per variable                   */
      argLength = atoi(contentLength);
      pHead = ReadArguments(argLength);
      pParm = pHead;

      /* Output the list of variable names and values */
      while (pParm)
        {
        printf("<PRE>Variable \"%s\" = %s</PRE><P>",
               pParm->VariableName,
               pParm->Value);

        pParm = pParm->pNext;
        }

      /* Output the remainder of the HTML used to display the results */
      printf("<P>\n");
      printf("<HR>\n");
       printf("</BODY>\n");
      printf("</HTML>\n");
      }
    }
  }
```

```
/*****************************************************************************/
/*                                                                           */
/* Function       : ReadArguments                                            */
/*                                                                           */
/* Description    : Read the arguments from stdin that are supplied          */
/*                  to a CGI program when the method is POST.                */
/*                  Breaks up the input into                                 */
/*                  (Variable, Value) pairs.                                 */
/*                  Handles translating of all the special characters        */
/*                  that HTTP puts into the strings.                         */
/*                                                                           */
/*****************************************************************************/
PPARMLIST ReadArguments(int InputLength)
  {
  PPARMLIST pCur= NULL;
  PPARMLIST pHead= NULL;
  PPARMLIST pPrev= NULL;
  char *Input;
  char *pToken;

  if (InputLength < 1)
    {
    return(NULL);
    }
  /* Allocate a buffer for the input */
  Input = malloc(InputLength + 1);

  if (Input == NULL)
    {
    return(NULL);
    }

/* Read the input */
  gets(Input);

  /* Variables are separated by the "&" character */
  pToken = strtok(Input, "&");

  while (pToken)
    {
    /* Create and fill in linked list of variable information */
    pCur = malloc(sizeof(PARMLIST));
    pCur->VariableName = pToken;
    pToken = strchr(pToken, '=');
    if (pToken)
      {
      *pToken = '\0';
      pCur->Value = ++pToken;
      PlusesToSpaces( pToken );
      TranslateEscapes( pToken );
 }
    else
      {
      pCur->Value = NULL;
      }
    if (pPrev)
       {
      pPrev->pNext = pCur;
      }

    if (!pHead)
      {
      pHead = pCur;
      }
    pPrev = pCur;
    pToken = strtok(NULL, "&");
```

```
    }

  if (pHead)
    {
    pPrev->pNext = NULL;
    }
  return(pHead);
  }

/*****************************************************************************/
/*                                                                           */
/* Function       : PlusesToSpaces (STATIC)                                  */
/*                                                                           */
/* Description    : This one's easy. It just translates any '+'              */
/*                  characters found into ' ' characters.                    */
/*                                                                           */
/*****************************************************************************/

  static void PlusesToSpaces(char *Str)
  {
  if (Str != NULL)
    {
    while (*Str != '\0')
 {
      if (*Str == '+')
        {
        *Str = ' ';
        }

      ++Str;
      }
    }
  }

 static int HexVal(char c)
  {
  int rc;

  switch (c)
    {
    case '1':
      rc = 1;
      break;
    case '2':
      rc = 2;
      break;

    case '3':
      rc = 3;
      break;

    case '4':
      rc = 4;
      break;

    case '5':
      rc = 5;
      break;

    case '6':
      rc = 6;
      break;

    case '7':
      rc = 7;
      break;
```

```
    case '8':
      rc = 8;
      break;

    case '9':
      rc = 9;
      break;

    case 'A':
    case 'a':
      rc = 10;
      break;

    case 'B':
    case 'b':
      rc = 11;
      break;


    case 'C':
    case 'c':
      rc = 12;
      break;

    case 'D':
    case 'd':
      rc = 13;
      break;

    case 'E':
    case 'e':
      rc = 14;
      break;

    case 'F':
    case 'f':
      rc = 15;
      break;

    default:
      rc = 0;
      break;
    }

  return(rc);
  }

/*****************************************************************************/
/*                                                                           */
/* Function       : TranslateEscapes (STATIC)                                */
/*                                                                           */
/* Description    : Translate the escape sequences induced by HTTP. The      */
/*                  sequences consist of %xx, where xx is a hex number.      */
/*                  We replace the % character with the actual character     */
/*                  (i.e., the one whose ASCII value is xx), and then        */
/*                  shift over the rest of the string to remove the xx.      */
/*                  This is done in-place.                                   */
/*                                                                           */
/*****************************************************************************/
static void TranslateEscapes(char *Str)
  {
  char *NextEscape;
  char ConvertValue [2];
  int AsciiValue;

  NextEscape = strchr(Str, '%');
  while (NextEscape != NULL)
```

```
    {
    AsciiValue = (16 * HexVal(NextEscape[1])) + HexVal(NextEscape[2]);
    ConvertValue[1] = '\0';
    ConvertValue[0] = (char) AsciiValue;
    __atoe(ConvertValue);  /*now it is an EBCDIC character */
    strncpy(NextEscape, ConvertValue, 1);
    memmove(&NextEscape[1], &NextEscape[3], strlen(&NextEscape[3]) + 1);
    NextEscape = strchr(&NextEscape[1], '%');
    }
  }

/*****************************************************************************/
/* This function will output a message if an error occurs when attempting     */
/* to display a HTML page.                                                    */
/*****************************************************************************/
int ErrMsg (char *msg)
  {
  printf("Content-type: text/html\n\n");
  printf("<HTML>\n");
  printf("<HEAD>\n");
  printf("<TITLE>Error</TITLE>\n");
  printf("</HEAD>\n");
  printf("<BODY>\n");
  printf("<H1>Error</H1>\n");
  printf("<HR>\n");
  printf("<P>\n");
  printf("An error occurred in the CGI program.\n");
  printf("The specific error message is shown below:\n");
  printf("<P>\n");
  printf("<PRE>%s</PRE>\n<P>", msg);
  printf("<P>\n");
  printf("<HR>\n");
  printf("</BODY>\n");
  printf("</HTML>\n");
  return(TRUE);
  }
```

CGI REXX program

The next example is written in REXX. It could process the form by calling the REXX EXEC CGIXMP.CMD; as shown in the example in "Creating HTML documents that reference CGI programs" on page 326. To do this, simply replace the line `<form method=POST action="cgi-bin/cgixmp">` in the HTML source with `<form method=GET action="cgi-bin/CGIXMP.CMD">`.

In this example, cgiutils and cgiparse commands are used to generate MIME type headers and to parse the HTTP request from the form.

Example

```
/* REXX */
'cgiutils -status 200 -ct text/html'
say '<HTML>'
say '<BODY>'
say '<H1>Processing forms with CGI scripts</H1>'
say '<P>'
say 'This is an example of a CGI script which uses the <CODE>'
say 'cgiutils</CODE> and <CODE>cgiparse</CODE>'
say 'functions to create a valid HTTP header and parse forms data,'
say 'respectively.'
say 'The <CODE>cgiutils</CODE> program is used to create'
say ' a set of HTTP'
say 'headers for the CGI script output.'
say '<P>'
say 'The <CODE>cgiparse</CODE> program is used to extract'
say ' data submitted as HTTP GET or POST requests from'
```

```
say 'forms The program parses variables from the CGI QUERY_STRING'
say 'environment variable.'
say 'The <CODE>cgiparse</CODE> program is useful for'
say 'extracting information such as the value of'
say 'a specific form field or the number of unique fields submitted'
say 'from a form.'
say 'This information can then be processed by CGI scripts in'
say 'performing their intended functions.'
say '<p>'
say 'Some examples of how the <CODE>cgiparse</CODE> program'
say ' can be used are shown below'
say '<HR>'
say '<H2>Parsing forms requests using <CODE>cgiparse'
say '</CODE>'
say '</H2>'
say ''
say 'The following is an example of the output that is generated by'
say 'using the -form option of the'
say '<CODE>cgiparse</CODE> command. The -form option'
say 'will parse the QUERY_STRING as a form request'
say 'and output the fields as a string of variables.'
say '<P>'
say 'The format of the <CODE>cgiparse</CODE> command is'
say '<CODE> cgiparse -form </CODE>.'
say '<P>'
say '<CODE>'
'cgiparse -form'
say'</CODE>'
say '<HR>'
say '<H2>Querying values using <CODE>cgiparse</CODE>'
say '</H2>'
say 'The following is an example of using the <CODE>cgiparse'
say '</CODE>'
say 'command to query the values of specific'
say 'fields parsed from the QUERY_STRING.'
say '<P>'
say 'The format of the <CODE>cgiparse</CODE> command is'
say '<CODE>cgiparse -value <CITE>fieldname</CITE>'
say '</CODE>'
say '<P>'
say 'Value for variable 1 = <CODE>'
'cgiparse -value var1'
say '<BR></CODE> '
say 'Value for variable 2 = <CODE>'
'cgiparse -value var2'
say '<BR></CODE>'
say 'Value for variable 3 = <CODE>'
'cgiparse -value var3'
say '<BR></CODE>'
say 'Value for variable 4 = <CODE>'
'cgiparse -value var4'
say '<BR></CODE>'
say 'Value for variable 5 = <CODE>'
'cgiparse -value var5'
say '<BR></CODE>'
say 'Value for variable 6 = <CODE>'
'cgiparse -value var6'
say '</CODE>'
say '<P>'
say '<HR>'
say '<H2>Determining the number of fields in QUERY_STRING using'
say '<CODE>cgiparse</CODE></H2>'
say 'The <CODE>cgiparse</CODE> function can also be used to'
say 'determine the number of unique fields that are submitted with'
say 'a form.'
say '<P>'
say 'The format of the <CODE>cgiparse</CODE> command is'
```

```
say '<CODE>cgiparse -form -count</CODE>.'
say '<P>'
say 'Number of unique fields = <CODE>'
'cgiparse -form -count'
say '<BR></CODE>'
say '<HR>'
say '</BODY>'
say '</HTML>'
```

CGI shell script

The next example is written in the shell scripting language. It can process the form shown in the example on page 327 by calling the shell script command CGIXMP.SH. To do this, simply replace the line `<form method=POST action="cgi-bin/cgixmp">` in the HTML source with `<form method=GET action="cgi-bin/CGIXMP.SH">`.

In this example, cgiutils and cgiparse commands are used to generate MIME type headers and to parse the HTTP request from the form.

Example

```
#!/bin/sh
# This simple shell script will take the arguments passed in,
# and print them out

cgiutils -status 200 -ct text/html
echo '<HTML>';
echo '<BODY>';
echo '<H1>Processing forms with CGI scripts</H1>';
echo '<P>';
echo 'This is an example of a CGI script which uses the <CODE>'
echo 'cgiutils'</CODE> and <CODE>cgiparse</CODE>'
echo 'functions to create a valid HTTP header and parse forms data,'
echo 'respectively.'
echo 'The <CODE>cgiutils</CODE> program is used to create'
echo ' a set of HTTP'
echo 'headers for the CGI script output.'
echo '<P>'
echo 'The <CODE>cgiparse</CODE> program is used to extract'
echo ' data submitted as HTTP GET or POST requests from'
echo 'forms The program parses variables from the CGI QUERY_STRING'
echo 'environment variable.'
echo 'The <CODE>cgiparse</CODE> program is useful for'
echo 'extracting information such as the value of'
echo 'a specific form field or the number of unique fields submitted'
echo 'from a form.';
echo 'This information can then be processed by CGI scripts in'
echo 'performing their intended functions.'
echo '<P>'
echo 'Some examples of how the <CODE>cgiparse</CODE> program'
echo ' can be used are shown below'
echo '<HR>';
echo '<H2>Parsing forms requests using <CODE>cgiparse'
echo '</CODE>echo '</H2>';
echo '
echo 'The following is an example of the output that is generated by'
echo 'using the -form option of the'
echo '<CODE>cgiparse</CODE>'echo 'command.  The -form option'
echo 'will parse the QUERY_STRING as a form request'
echo 'and output the fields as a string of variables.'
echo '<P>'
echo 'The format of the <CODE>cgiparse</CODE> command is'
echo '<CODE> cgiparse -form </CODE>.' ;
echo '<P>'
echo '<CODE>'
```

```
giparse -form
echo'</CODE>'
echo '<HR>';
echo '<H2>Querying values using <CODE>cgiparse</CODE>'
echo '</H2>'
echo 'The following as an example of using the <CODE>cgiparse'
echo '</CODE>'
echo 'command to query the values of specific'
echo 'fields parsed from the QUERY_STRING.'
echo '<P>'
echo 'The format of the <code>cgiparse</CODE> command is'
echo '<CODE>cgiparse -value <CITE>fieldname</CITE>'
echo '</CODE>'
echo '<P>'
echo 'Value for variable 1 = <CODE>'
cgiparse -value var1
echo '<BR></CODE> '
echo 'Value for variable 2 = <CODE>'
cgiparse -value var2
echo '<br></CODE>'
echo 'Value for variable 3 = <CODE>'
cgiparse -value var3
echo '<BR></CODE>'
echo 'Value for variable 4 = <CODE>'
cgiparse -value var4
echo '<BR></CODE>'
echo 'Value for variable 5 = <CODE>'
cgiparse -value var5
echo '<BR></CODE>'
echo 'Value for variable 6 = <CODE>'
cgiparse -value var6
echo '</CODE>'
echo '<P>'
echo '<HR>'
echo '<H2>Determining the number of fields in QUERY_STRING using'
echo '<CODE>cgiparse</CODE></H2>'
echo 'The <CODE>cgiparse</CODE> function can also be used to'
echo 'determine the number of unique fields that are submitted with'
echo 'a form.'
echo '<P>'
echo 'The format of the <CODE>cgiparse</CODE> command is'
echo '<CODE>cgiparse -form -count</CODE>.'
echo '<P>'
echo 'Number of unique fields = <CODE>'
cgiparse -form -count
echo '<BR></CODE>'
echo '<HR>'
echo '</BODY>'
echo '</HTML>'
```

Chapter 19. Writing GWAPI programs

Overview of the GWAPI. 345
General procedure for writing GWAPI programs 346
GWAPI samples 346
Storing GWAPI programs 346
Guidelines for writing GWAPI programs . . . 346
Server request process 347
Process steps for the Web server 348
Application functions. 349
Application function prototypes 349
HTTP return codes and values 353
Predefined functions and macros 359
Return codes from predefined functions and macros 359
GWAPI configuration directives 360
GWAPI usage notes 361
GWAPI directives and syntax 361
GWAPI directive variables 361
Compatibility with other APIs. 362
Porting CGI programs 362
GWAPI reference information 363
Authentication and authorization. 363
Environment variables 365
HTCodePage_t, the other data type 366
z/OS GWAPI REXX applications 367
Invoking REXX executable programs as GWAPI applications 367
Writing GWAPI REXX Executable Programs . . 369
GWAPI REXX executable program conditions 369
Calling external subroutines and functions 370
Debugging and problem determination. . . 370
Transience of REXX variables 370
Calling GWAPI predefined functions . . . 370
Link routines to GWAPI functions 371
GWAPI REXX executable program exit conditions: 373
Example GWAPI REXX service executable program 373
GWAPI REXX downloads 374
Example GWAPI REXX data filter executable program 374
Debugging C/C++ GWAPI programs 375
Overview of support and restrictions 375
Getting started 376
Step 1. Install the remote debugger 376
Step 2. Compile your C/C++ program with the TEST compile-time option 376
Step 3. Edit the Web server configuration file 376
Step 4. Start the remote debugger on the workstation 377
Step 5. Start or restart the Web server . . . 377
Troubleshooting hints and tips 377
Debug Tool hints and tips 377
Web server error messages 377

Overview of the GWAPI

The Go Webserver Application Programming Interface (GWAPI) is an interface to the HTTP Server that allows you to extend the server's base functions. You can write extensions to do customized processing, such as:

- Enhance the basic authentication or replace it with a site-specific process
- Add error handling routines to track problems or alert for serious conditions
- Detect and track information that comes in from the requesting client, such as server referrals and user agent code

The GWAPI compares favorably to other similar APIs (such as Netscape's NSAPI and Microsoft's ISAPI) for these reasons:

- **Efficiency**
  - Designed specifically for the threaded (rather than forked) processing used by the HTTP Server.
- **Flexibility**
  - Contains rich and compatible functions that can be used to clone other APIs.
  - Is platform independent and language neutral. It runs on all the Web server platforms, and applications can be written in any of the programming languages supported by these platforms.
- **Ease of use**
  - Passes simple data types by reference, not value (for example, long * or char *)
  - Has a fixed number of parameters for each function

- Has no return values from functions but does have a return code parameter
- Includes bindings for the C language
- Does not impact allocated memory (the GWAPI does not free the user's memory and the user does not free the GWAPI's memory)

General procedure for writing GWAPI programs

Before you start writing your GWAPI programs, you need to understand how the HTTP Server works. The server performs a sequence of steps for each client request that it processes. Your program can execute and make function calls at any of these steps. Decide where in the basic server request process you want to add customized functions. For example, do you want to do something after a client request is read but before performing any other processing? Or, maybe you want to perform special routines during authentication and then again after the requested file is sent.

Logically, your program will have two parts: application functions (written by you) to handle one of the steps and predefined functions (provided with the GWAPI) that you call to manipulate the request. You can instruct the server to call the application functions in your program at the appropriate processing steps by using the GWAPI directives in your server configuration file.

GWAPI samples

To help you get started with your own GWAPI functions, look at the sample programs on the Web site. You can access these samples at URL:

`http://www.ibm.com/software/webservers/httpservers/doc53.html`

Storing GWAPI programs

You can mark your GWAPI programs Program Controlled and store them in an HFS.

Guidelines for writing GWAPI programs

1. Write your program, following the syntax and guidelines we provide for the server's application functions. Give each of your application functions a unique function name and call the server predefined functions as needed.
2. Be sure to include HTAPI.h and to use the HTTPD_LINKAGE macro in your function definitions to avoid abending the server. This macro ensures that all the functions use the same calling conventions.
3. The server runs in a multi-threaded environment; therefore, your application functions **must** be threadsafe. If your application is re-entrant, performance will not decrease.
4. You can write multi-threaded GWAPI applications for the Web server. This support enables GWAPI applications to create more threads if needed. Note that the GWAPI application is responsible for creating and managing new threads. If the application needs more threads, a thread create is required. The maximum number of threads that can be created is determined by the system limit settings for MAXTHREADS and MAXTHREADTASKS in the SYS1.PARMLIB(BPXPRMxx) member.

   Important: The Web server requires that you design your multi-threaded GWAPI applications to use only the base Web server thread for communications, for example, calling HTTPD_write. The base thread is the thread under which the Web server was running when it invoked the GWAPI. Using other threads with the Web server's predefined functions and macros,

such as HTTPD_write or HTTPD_set, may cause unpredictable results which include causing the Web server to crash.

5. **Keep the actions in your applications to a thread scope.** Do **not** perform any actions at a process scope, such as exit, change user ID, registering signal handlers, etc.
6. Eliminate global variables or protect them with a mutual exclusion semaphore.
7. Do not forget to set the Content-Type header if you are using HTTPD_write() to send data back to the client.
8. You will also need to take into consideration the Content-Encoding header when you use HTTPD_write() to send data back to the client. This is especially important for the z/OS system where the native code page is EBCDIC.

   For example, if the data is stored on z/OS in the EBCDIC code page, you must use HTTPD_set() to set the CGI environment variable CONTENT_ENCODING to the appropriate code page (in this case, to "ebcdic") before you use HTTPD_write() to send the data out to the client. If you fail to do this, the client will display unreadable content.
9. Always check your return codes and provide conditional processing where necessary.
10. A GWAPI program must be packaged as a DLL. Compile and link your program, referring to the documentation for the C compiler to build a DLL or .so file. The following example can be used as a guideline:

    ```
    c89 -W"c,dll,expo" -W"l,dll" -I /usr/lpp/internet/samples/API -o pwapi.so
    pwapi.c /usr/lpp/internet/samples/API/libhttpdapi.x
    ```

    - /usr/lpp/internet/samples/API contains the header files.
    - /usr/lpp/internet/samples/API/libhttpdapi.x is the side-deck defining the GWAPI API functions to the binder.

    Refer to the *z/OS C/C++ Programming Guide* for information on building and using DLLs. Refer to the *z/OS UNIX System Services Command Reference* for information on the c89 command.
11. Add GWAPI directives to your configuration file so that you can associate your program's application functions with the appropriate step. There is a separate directive for each server request processing step. Stop and restart your server to make the new directives take effect.
12. Test your program rigorously. Because the server is a threaded server, you should apply more rigorous testing than you would for a forking server. Because the server calls your program directly and they both run in the same process space, errors in your program can crash the server.

Server request process

The basic server request process can be broken up into a number of steps, based on the type of processing the server is performing during that phase. Each step includes a juncture at which a specified part of your program can execute. By adding GWAPI directives to your configuration file, you indicate which of your application functions you want called during a particular step. You can call several application functions during a request process step by including more than one of the GWAPI directives for that step.

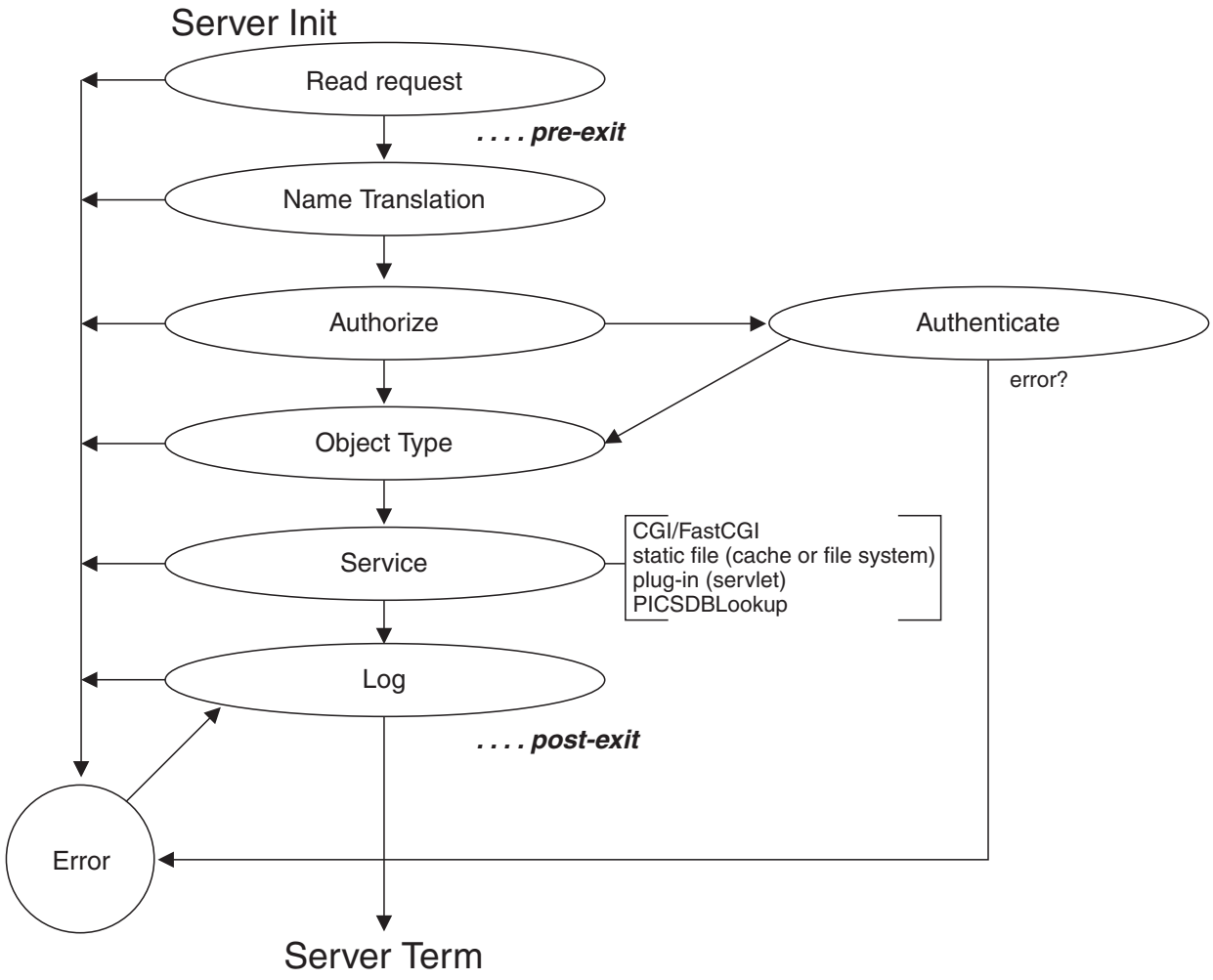
Your compiled program resides in a DLL or .so file, depending on your operating system. As the server proceeds through its request process steps, it calls the application functions associated with each step, until one of the functions indicates

it has handled the request. If you have more than one of your application functions indicated for a particular step, they are called in the order in which they appear in the configuration file.

If the request is not handled by an application function (either you did not include a GWAPI directive or your application function for that step returned HTTP_NOACTION), the server will perform its default action for that step.

Note: This is true for all steps except the Service step; the Service step does not have a default action.

Process steps for the Web server



The following list indicates the purpose of each step and defines the processing order.

Server Initialization
Performs initialization functions before any client requests are read.

WLMClassify
When the Web server is running in Scalable Server mode, performs processing after a request is read but before the work is put on the work queue. If this pre-exit step is used, the user ID defined in the WLM policy is modified. For more information, see "WLMClassify - Customize the WLM pre-exit step" on page 478.

PreExit
Performs processing after a request is read but before anything else is done.

If this step returns an indication that the request was processed (HTTP_OK), skips the other steps in the request process and performs only the Data Filter, Log, and PostExit steps.

Authentication
Decodes, verifies, and stores security tokens. See the "Authentication and authorization" on page 363 for more information.

Name Translation
Translates the virtual path (from URL) to the physical path.

Authorization
Uses stored security tokens to check the physical path (protections, acls) and generates the WWW-Authenticate headers required for basic authentication. If you write your own application function to replace this step, you must generate these headers yourself. See "Authentication and authorization" on page 363 for more information.

Object Type
Locates the filesystem object indicated by the path.

Service
Satisfies the request (send the file, run the CGI, etc.)

PICSDBLookup
Looks for the PICS labels for the specified URL.

Data Filter
Gives write access to the outgoing data stream.

Log Allows transaction logging.

Error Allows customized responses to error conditions.

PostExit
Allows ″cleanup″ of resources allocated for request processing.

Server Termination
Allows ″cleanup″ processing when an orderly shutdown or restart occurs.

Application functions

Follow the syntax presented in the "Application function prototypes" to write your own program functions for the defined request processing steps.

Each of your functions must fill in the return code parameter with a value that indicates what action was taken.

- HTTP_NOACTION (value of `0`) means no action was taken.
- Otherwise one of the valid HTTP return codes is expected, indicating the application function handled the step. As a result, no other application functions will be called to handle that step of this request.

Application function prototypes

The function prototypes for each request step show the format to use and explain the type of processing they can perform. Note that the function names are not predefined. You must give your functions unique names and can choose your own naming conventions. For ease of association, we use names that relate to the server's request processing steps.

Server Initialization

```
void
HTTPD_LINKAGE ServerInit(
     unsigned char *handle, unsigned long *major_version,
     unsigned long *minor_version, long *return_code);
```

Called once when your module is loaded during server initialization, it is your opportunity to perform initialization before any requests have been accepted. Although all server initialization functions are called, error return codes from this step will cause the server to ignore all other functions configured in this program.

WLMClassify

```
void
HTTPD_LINKAGE WLMClassify(
    unsigned char *handle, long *return_code);
```

When the Web server is running in Scalable Server mode, called after the request has been read but before any processing has occurred. Used to modify the user ID defined in a WLM policy. Change the user ID by setting the CLASSIFY_USERID environment variable. The Web server first sets the CLASSIFY_USERID during the WLMClassify step to the user ID that comes from a client.

PreExit

```
void
HTTPD_LINKAGE PreExit(
    unsigned char *handle, long *return_code);
```

Called after the request has been read, but before any processing has occurred.

All server predefined functions are valid during this step.

Authentication

```
void
HTTPD_LINKAGE Authentication(
    unsigned char *handle, long *return_code);
```

Allows user verification of the security tokens. This step is performed based on the authentication scheme.

Only HTTPD_extract(), HTTPD_set(), HTTPD_authenticate(), and HTTPD_local_security are valid during this step.

Name Translation

```
void
HTTPD_LINKAGE NameTrans(
    unsigned char *handle, long *return_code);
```

Provides a mechanism for mapping URLs to objects.

Only HTTPD_extract(), HTTPD_set(), and HTTPD_authenticate() are valid during this step.

Authorization

```
void
HTTPD_LINKAGE Authorization(
    unsigned char *handle, long *return_code);
```

Verifies that the identified object may be returned to the client.

Only HTTPD_extract() and HTTPD_set() are valid functions during this step.

Object Type

```
void
HTTPD_LINKAGE ObjType(
    unsigned char *handle, long *return_code);
```

Checks to see if the object exists and performs object typing.

Only HTTPD_extract() and HTTPD_set() are valid during this step.

Service

```
void
HTTPD_LINKAGE Service(
    unsigned char *handle, long *return_code);
```

Satisfies the request, if not satisfied in the PreExit.

All server predefined functions are valid during this step. For information on configuring your Service function to execute based on the HTTP method rather than the URL, see the description of the Enable directive in "Enable - Enable HTTP methods" on page 519.

PICSDBLookup

```
void
HTTPD_LINKAGE PICSDBLookup(
    unsigned char *handle, long *return_code);
```

Allows you to either dynamically create a PICS label for the requested document or to search for a PICS label in an alternative file or database. Your application function can evaluate the PICS_SERVICENAME, PICS_SITENAME, and PICS_PATHNAME variables to determine which document was requested.

If your application function finds or creates a label, it must call the HTTPD_supply_label() predefined function to make the label available to the Web server and return a code of 200. Only HTTPD-extract(), HTTPD_set() and HTTPD_supply_label() are valid during this step.

Note: You can process this step more than once for each request. If a request asks for labels for multiple documents or from multiple rating services, the server will call the application function once for each label that is requested.

Data Filter

Filters data as a "stream class", which means that each of its functions acts like a segment of pipe down which data flows. For this step, you must implement three application functions:

```
void
HTTPD_LINKAGE open(
    unsigned char *handle, long *return_code);
```

Performs any initialization (such as buffer allocation) required to process the data for this stream. An error return code will cause this filter to abort (the write and close functions will not be called).

```
void
HTTPD_LINKAGE write(
    unsigned char *handle, unsigned char *data,
    unsigned long *length, long *return_code);
```

Or,

```
void
HTTPD_LINKAGE write(
    unsigned char *handle, unsigned char *data,
    unsigned long *length, long *return_code,
    HTCodePage_t *codepage);
```

Processes the data and calls the server's write function with the new or changed data. The application must not attempt to free the buffer passed to it nor expect the server to free the buffer it receives.

The HTCodePage_t parameter is available for the HTTP Server Version 4.6.1 and later. GWAPI users cannot use the `DataFilter`'s compiled using this implementation with previous versions of the Web server. The earlier versions will not support the additional codepage parameter and unexpected behavior will result.

```
void
HTTPD_LINKAGE HTTPD_writeCP(
     unsigned char *handle, unsigned char *value, unsigned long *value_length,
     HTCodePage_t *code_page, long *return_code);
```

The code page specification allows the GWAPI user to specify the body code page. To view the definition for the HTCodePage_t data type see "HTCodePage_t, the other data type" on page 366.

```
void
HTTPD_LINKAGE close(
    unsigned char *handle, long *return_code);
```

Performs any cleanup (such as flushing and freeing the buffer) required to complete processing the data for this stream.

Note: Be aware that undesirable behavior may occur if data filtering applications are not properly selective on their filtering of data streams. CGI's may not work if filtered incorrectly, GIF files may not display, and other binary streams may not work as expected.

Log

```
void
HTTPD_LINKAGE Log(
    unsigned char *handle, long *return_code);
```

Called after each request has been processed and the communication to the client has been closed, regardless of the success or failure of the request processing. Only HTTPD_extract() and HTTPD_set() are valid during this step.

Error

```
void
HTTPD_LINKAGE Error(
    unsigned char *handle, long *return_code);
```

Called only when an error is encountered and provides an opportunity to customize the response.

PostExit

```
void
HTTPD_LINKAGE PostExit(
    unsigned char *handle, long *return_code);
```

Called, regardless of the success or failure of the request, so that you can clean up any resources allocated by your application to process the request.

Server Termination

```
void
HTTPD_LINKAGE ServerTerm(
    unsigned char *handle, long *return_code);
```

This function is processed when an orderly shutdown or restart of the server occurs. It allows you to clean up resources allocated during the Server Initialization step. Do not call any HTTP_* functions in this step (the results are unpredictable). If you have more than one GWAPI directive in your configuration file for Server Termination, they will all be called.

HTTP return codes and values

These return codes follow the HTTP specification published by the World Wide Web Consortium at URL:

`http://www.w3.org/Protocols/`

Your application functions should return one of the following values:

Value	Return code
0	HTTP_NOACTION
100	HTTP_CONTINUE
101	HTTP_SWITCHING_PROTOCOLS
200	HTTP_OK
201	HTTP_CREATED
202	HTTP_ACCEPTED
203	HTTP_NON_AUTHORITATIVE
204	HTTP_NO_CONTENT
205	HTTP_RESET_CONTENT
206	HTTP_PARTIAL_CONTENT
300	HTTP_MULTIPLE_CHOICES
301	HTTP_MOVED_PERMANENTLY
302	HTTP_MOVED_TEMPORARILY
303	HTTP_SEE_OTHER
304	HTTP_NOT_MODIFIED
305	HTTP_USE_PROXY
400	HTTP_BAD_REQUEST
401	HTTP_UNAUTHORIZED
403	HTTP_FORBIDDEN
404	HTTP_NOT_FOUND
405	HTTP_METHOD_NOT_ALLOWED
406	HTTP_NOT_ACCEPTABLE
407	HTTP_PROXY_UNAUTHORIZED

Value	Return code
408	HTTP_REQUEST_TIMEOUT
409	HTTP_CONFLICT
410	HTTP_GONE
411	HTTP_LENGTH_REQUIRED
412	HTTP_PRECONDITION_FAILED
413	HTTP_ENTITY_TOO_LARGE
414	HTTP_URI_TOO_LONG
415	HTTP_BAD_MEDIA_TYPE
500	HTTP_SERVER_ERROR
501	HTTP_NOT_IMPLEMENTED
502	HTTP_BAD_GATEWAY
503	HTTP_SERVICE_UNAVAILABLE
504	HTTP_GATEWAY_TIMEOUT
505	HTTP_BAD_VERSION

Predefined functions and macros

You can call the server's predefined functions and macros from your own application functions. You must use their predefined names and follow the format we describe below. Note that the parameter descriptions use the letter ″i″ to indicate input, ″o″ to indicate output, and ″i/o″ to indicate that a parameter is both input and output.

Each of these functions will return one of the HTTPD return codes, depending on the success of the request. Use the handle as the first parameter when calling these functions; Otherwise, the function will return a HTTPD_PARAMETER_CHECK error code. NULL is not accepted as a valid handle.

HTTPD_authenticate()

Authenticates a userid/password. Valid only in PreExit, Authenticate, and Authorization steps.

```
void
HTTPD_authenticate(
            unsigned char *handle,   /* i; handle */
            long *return_code);      /* o; return code */
```

This function returns HTTPD_AUTHENTICATE_FAILED if the authenticate fails.

HTTPD_attributes

Extracts the attributes of a file. Valid in all steps.

```
void
HTTPD_attributes(
  unsigned char *handle,       /* i; handle */
  unsigned char *name,         /* i; name of the file */
  unsigned long *name_length,  /* i; length of the name */
  unsigned char *value,        /* o; buffer containing attributes*/
  unsigned long *value_length,/* i/o;size of buffer/length */
                               /*   of attributes*/
  long *return_code);         /* o; return code */
```

HTTPD_extract()

Extracts the value of a variable associated with this request. The valid variables you can use for the `name` parameter are the same as those used in the CGI. See "Environment variables" on page 365 for more information. This function is valid in all steps, though not all variables will be.

```
void
HTTPD_extract(
 unsigned char *handle,       /* i; handle */
 unsigned char *name,         /* i; name of value to extract */
 unsigned long *name_length,  /* i; length of the name */
 unsigned char *value,        /* o; buffer in which to put value */
 unsigned long *value_length, /* i/o; buffer size/length of value */
          long *return_code); /* o; return code */
```

If this function returns the HTTPD_BUFFER_TOO_SMALL return code, the buffer size you requested was not big enough for the extracted value. In this case, the function does not fill in the buffer but does update `value_length` with the buffer size you would need in order to successfully extract this value. Retry the extract with a buffer that is at least as big as the returned `value_length`.

HTTPD_reverse_translate()

translates a file system path to a URL. Valid in all steps.

```
void
HTTPD_reverse_translate(
 unsigned char *handle,       /* i; handle */
 unsigned char *name,         /* i; name of the file system object*/
 unsigned long *name_length,  /* i; length of the name */
 unsigned char *value,        /* o; buffer which contains the URL */
 unsigned long *value_length, /* i/o; size of buffer/length of URL*/
          long *return_code); /* o; return code */
```

HTTPD_translate()

translates a URL to a file system path. Valid in all steps.

```
void
HTTPD_translate(
    unsigned char *handle,       /* i; handle  */
    unsigned char *name,         /* i; name of the URL */
    unsigned long *name_length,  /* i; length of the name */
    unsigned char *url_value,    /* o; buffer containing
translated URL */
    unsigned long *url_value_length,
      /* i/o; buffer size/length of translated URL */
    unsigned char *path_trans,   /* o; buffer containing
PATH_TRANSLATED*/
    unsigned long *path_trans_length, /* i/o; size of
buffer/length of PATH_TRANSLATED*/
    unsigned char *query_string, /* o; buffer containing
QUERY_STRING*/
    unsigned long *query_string_length, /* i/o;size of
buffer/length of QUERY_STRING*/
             long *return_code); /* o; return code */
```

HTTPD_log_access()

writes a string to the server's access log. When writing a message to the access log, it is not necessary to use the escape character when printing the percent (%) character.

```
void
HTTPD_log_access(
    unsigned char *handle,       /* i; handle */
```

```
    unsigned char *value,        /* i; data to write */
    unsigned long *value_length, /* i; length of the data */
             long *return_code); /* o; return code */
```

HTTPD_set()

Sets the value of a variable associated with this request. The valid variables you can use for the `name` parameter are the same as those used in the CGI. See "Environment variables" on page 365 for more information.

Note that you can also create variables with this function. If any variables you create are prefixed by "HTTP_", they will be sent as headers in the response, without the "HTTP_" prefix. For example, if you want to see a Location header, use HTTPD_set() with the variable name HTTP_LOCATION.

You can use this function to send Content-Encoding headers in the response. See "Response generation" on page 334 in the CGI chapter for more information on the Content-Encoding header and z/OS code page translation.

This function is valid in all steps, though not all variables will be.

```
void
HTTPD_set(
  unsigned char *handle,       /* i; handle */
  unsigned char *name,         /* i; name of the value to set */
  unsigned long *name_length,  /* i; length of the name */
  unsigned char *value,        /* i; buffer containing the value */
  unsigned long *value_length, /* i; length of value */
           long *return_code); /* o; return code */
```

HTTPD_supply_label

Provides the HTTP Server with a PICS label that was generated or retrieved during the PICSDBLookup step.

Valid only during the PICSDBLookup step.

```
void
HTTPD_supply_label(
    unsigned char *handle,           /* i; handle */
    unsigned char *value,             /* i; data to send */
    unsigned long *value_length,      /* i; length of the data */
             long *return_code);     /* o; return code */
```

HTTPD_file()

Sends a file to satisfy this request. Valid only in PreExit, Service, NameTrans, Error, and DataFilter steps.

```
void
HTTPD_file(
    unsigned char *handle,           /* i; handle */
    unsigned char *name,             /* i; name of file to send */
    unsigned long *name_length,      /* i; length of the name */
             long *return_code);     /* o; return code */
```

HTTPD_exec()

Executes a script to satisfy this request. Valid in PreExit, Service, NameTrans, and Error steps.

```
void
HTTPD_LINKAGE
HTTPD_exec(
    unsigned char *handle,       /* i; handle */
    unsigned char *name,         /* i; name of script to execute */
    unsigned long *name_length,  /* i; length of the name */
             long *return_code); /* o; return code */
```

HTTPD_read()

Reads the body of the client's request. Uses HTTPD_extract for headers. Valid only in the PreExit, Service and Data Filter steps.

Migration note: On the z/OS system, your applications no longer need to take the additional step to translate the data from ASCII to EBCDIC. If you are migrating from a previous version of the HTTP Server, check your GWAPI functions for this processing.

```
void
HTTPD_read(
 unsigned char *handle,       /* i; handle */
 unsigned char *value,        /* i; buffer in which to place data */
 unsigned long *value_length, /* i/o; buffer size/length of data */
          long *return_code); /* o; return code */
```

HTTPD_write()

Writes the body of the response. Uses HTTPD_set and HTTPD_extract for headers. Valid in the PreExit, Service, NameTrans, Error, and Data Filter steps.

If you do not use HTTPD_set() to set the content type before the first time you call this function, the server will assume you are sending a CGI data stream.

You may need to use HTTPD_set() to set the CGI environment variable CONTENT_ENCODING to the appropriate code page for the content of your response (for example "ebcdic") before you send the data to the client.

```
void
HTTPD_write(
    unsigned char *handle,            /* i; handle */
    unsigned char *value,             /* i; data to send */
    unsigned long *value_length,      /* i; length of the data */
             long *return_code);      /* o; return code */
```

HTTPD_log_error()

Writes a string to the server's error log.

```
void
HTTPD_LINKAGE
HTTPD_log_error(
    unsigned char *handle,       /* i; handle */
    unsigned char *value,        /* i; data to write */
    unsigned long *value_length, /* i; length of the data */
             long *return_code); /* o; return code */
```

Attention: Plug-in writers do not need to escape their percent (%) signs when using HTTPD_log_error. Previous releases use the string as a printf format string which required you to escape (%%) any percent characters in the error string.

HTTPD_log_trace()

Writes a string to the server's trace log. HTTPD_log_trace() uses printf format strings to write to the trace log. If you include a percent (%) character in the message, you must escape it using %%.

```
unsigned char *value="CPU utilization at 100%%";
unsigned long value_length=strlen(value);
long return_code;

HTTPD_log_trace(handle, value, &value_length, &return_code);
```

The above example will display the message *CPU utilization at 100%*.

```
void
HTTPD_LINKAGE
HTTPD_log_trace(
    unsigned char *handle,       /* i; handle */
    unsigned char *value,        /* i; data to write */
    unsigned long *value_length, /* i; length of the data */
             long *return_code); /* o; return code */
```

HTTPD_restart()

Restarts the server after all active requests have been processed. Valid in all steps EXCEPT for ServerInit, ServerTerm, and Datafilter.

```
void
HTTPD_restart(
             long *return_code);      /* o; return code */
```

HTTPD_proxy()

Makes a proxy request. Valid in PreExit and Service steps.

Note: This is a completion function; the response is complete after this function.

```
void
HTTPD_LINKAGE
HTTPD_proxy(
    unsigned char *handle,       /* i; handle */
    unsigned char *url_name,     /* i; url for the proxy request */
    unsigned long *name_length,  /* i; length of the url */
    unsigned char *request_body, /* i; body of the request */
    unsigned long *body_length,  /* i; length of the body */
              long *return_code); /* o; return code */
```

HTTPD_local_security()

This function establishes a local host (RACF) user ID based on an application name (Lotus Notes) and user ID. The system default authorization step establishes the local security environment based on configured Protect rules and the local user ID. The local (RACF) user ID is obtained by calling the RACF Generic ID Mapping function. This function is valid ONLY in the authentication step.

For detailed steps on activating the Lotus Notes Adapter function for HTTPD_local_security() to work properly, see the installation steps in "Lotus Notes access using RACF user IDs" on page 28. Also, see z/OS Security Server (RACF) Callable Services or Macros and Interfaces documentation for RACF function details.

```
void
HTTPD_local_security
     (unsigned char *handle,
      unsigned char *appl_userid,       /* Notes short name */
      unsigned long *appl_userid_len,
      unsigned char *appl_name,         /* NOTES */
      unsigned long *appl_name_len,
                long *reason_code,
                long *return_code)
```

Note: The application's GWAPI is responsible to verify the application user ID and password are authentic before calling HTTPD_local_security. HTTPD_local_security does not authenticate an application user ID or password; it only retrieves and sets the associated RACF user ID if one exists.

The server will fill in the reason code parameter with one of these values depending on the success of the HTTPD_local_security() request.

Reason Code Value	Return Code Value/Explanation
1	HTTPD_SUCCESS The local user ID is correct.
2	HTTPD_FAILURE Generic ID Mapping failed with one of the following RACF return codes: 16, 24, 28, 32 (no user ID mapping available).
3	HTTPD_FAILURE Generic ID Mapping failed with a RACF return code of 20. Not authorized to use this service.
4	HTTPD_FAILURE Generic ID Mapping failed with other RACF return codes. Internal parameter list error or RACF error.
5	HTTPD_Failure Internal error setting local user ID. Contact the IBM Software Support Center.

Note: Once an HTTPD_function returns, it is safe for you to free any memory you passed with it.

Return codes from predefined functions and macros

The server will fill in the return code parameter with one of these values depending on the success of the request.

Value	Status/Explanation
-1	HTTPD_UNSUPPORTED The function is not supported.
0	HTTPD_SUCCESS The function succeeded and the output fields are valid.
1	HTTPD_FAILURE The function failed.
2	HTTPD_INTERNAL_ERROR Encountered an internal error and cannot continue processing this request.
3	HTTPD_PARAMETER_ERROR One or more invalid parameters were passed. For example, the variable you tried to extract is unknown.
4	HTTPD_STATE_CHECK The function is not valid in this step.
5	HTTPD_READ_ONLY Returned only by HTTPD_set(). The variable is read-only and cannot be set by the application.

Value	Status/Explanation
6	HTTPD_BUFFER_TOO_SMALL Returned only by HTTPD_extract(). The provided buffer was too small.
7	HTTPD_AUTHENTICATE_FAILED Returned only by HTTPD_authenticate(). Examine the HTTP_RESPONSE and HTTP_REASON variables for more information.
8	HTTPD_EOF Returned only by HTTPD_read(). Indicates the end of the request body.
9	HTTPD_ABORT_REQUEST The request has been aborted because the client has provided an entity tag that did not match the condition specified by the request.
10	HTTPD_REQUEST_SERVICED Returned by HTTPD_proxy. Indicates the function that was called completed the response for this request.
11	HTTPD_RESPONSE_AREADY_COMPLETED The function failed because the response for that request has already been completed.
12	HTTPD_ALREADY_CLASSIFIED The function failed because the WLM classification has already completed. The reason the WLM classification has completed is one of the following: • The server is running in scalable mode. • The classification has already been issued within this request.
22	HTTPD_BAD_TOKEN_NONWORKER_THREAD The handle was corrupted and pointed to a storage area that did not contain the normal eye-catcher text.
23	HTTPD_BAD_TOKEN_WORKER_THREAD The handle was corrupted, and its first field contained a value that was out of range.
24	HTTPD_PGM_NOT_IN_A_PLUGIN The GWAPI service was called, but the thread was not called as a GWAPI exit.
25	HTTPD_ALREADY_IN_A_CALLBACK The GWAPI service was called, but the thread was already in a GWAPI function.

GWAPI configuration directives

Each step in the request process has a configuration directive that allows you to indicate which of your application functions you want called and executed during that step. You can add these directives to your server's configuration file by manually editing and updating it or by using the GWAPI Request Processing form in the server's Remote Configuration and Administration forms.

GWAPI usage notes

- When appropriate, you can indicate that you want your application function called for all URL requests or only for URL requests that match a specified mask.
- You can also have your Authentication functions called for every request or just for those with a type of Basic.
- The GWAPI directives, except for the Service and NameTrans directives, can be in any order in the configuration file and you do not need to include every one. If you do not have an application function for a particular step, just omit the corresponding directive.
- The Service and NameTrans directives work like the Exec directive and are dependent on its occurrence and placement relative to other mapping directives within the configuration file. This means that the server processes the Service, NameTrans, Map, Pass, Exec, Redirect, and Fail directives in their sequential order within the configuration file. When it successfully maps a URL to a file, it does not read or process any other of these directives.
- You can also have more than one configuration directive for a step. For example, you could include two NameTrans directives, each pointing to a different application function. When the server performs the name translation step, it will process your name translation functions in the order in which they appear within the configuration file.
- Multiple IP configuration (using a trailing IP address or template) is supported **only** for the Service and NameTrans directives.
- If the server fails to load a specific application function or you have a ServerInit directive that does not return an OK return code, no other application functions for that compiled GWAPI program will be called and any processing specific to that program which was done up to this point will be ignored. Other GWAPI programs that you include in these directives, and their application functions, will not be affected.

GWAPI directives and syntax

ServerInit		*/path/file:function_name init_string*
WLMClassify		*/path/file:function_name*
PreExit		*/path/file:function_name*
Authentication	*type*	*/path/file:function_name*
NameTrans	*/URI*	*/path/file:function_name IP_address_template*
Authorization	*/URI*	*/path/file:function_name*
ObjectType	*/URI*	*/path/file:function_name*
Service	*/URI*	*/path/file:function_name* IP_address_template*
Data Filter		*/path/file:function_name:function_name:function_name*
PICSDBLookup		*/path/file:function_name*
Log	*/URI*	*/path/file:function_name*
Error	*/URI*	*/path/file:function_name*
PostExit		*/path/file:function_name*
ServerTerm		*/path/file:function_name*

GWAPI directive variables

The variables in these directives have the following meanings:

type Used only with the Authentication directive. Determines if your application function will be called. Valid values are:

Basic Application function is called only for basic authentication requests

***** Application function is called for all requests

/URI Determines for which URIs your application function will be called. URI specifications in these directives are virtual (they do not include the protocol) but are preceded by a slash (/). For example, /www.ics.raleigh.ibm.com is correct but http://www.ics.raleigh.ibm.com is not. Valid values are:

A specific URI
Application function is called only for that URI

URI template
Application function is called only for URIs that match the template. You can specify a template as /URI*, /*, or *.

Note: A URI template is **required** with the Service directive if you want path translation to occur.

/path/file
The fully qualified file name of your compiled program

:function_name
The name you gave your application function within your program

In the DataFilter directive, you must supply the names of the open, write, and close functions.

The Service directive requires an asterisk (*) after the *:function_name*, if you want to have access to path information.

init_string
Optional on the ServerInit directive, this can contain any arbitrary text you want to pass to your application function. Use HTTPD_extract() to extract the text from the INIT_STRING variable.

IP_address_template
Used only with the Service and NameTrans directives on servers that have more than one IP address. Determines if your application function will be called only for requests coming in on a specific IP address or on a range of IP addresses.

Compatibility with other APIs

The GWAPI is compatible with other APIs, such as CGI. You can run your existing CGI programs on all the server's operating systems.

Porting CGI programs

Here are a few guidelines for porting CGI applications written in C to use the GWAPI:

1. Remove your main() entrypoint or rename it so you can build a DLL.
2. Eliminate global variables or protect them with a mutual exclusion semaphore.
3. Change the following calls in your programs:
   - Change printf() header calls to HTTPD_set().
   - Change printf() data calls to HTTPD_write().

- Change getenv() calls to HTTPD_extract(). Note, this returns unallocated memory, so you must free the result.

4. Remember, the server runs in a multi-threaded environment and your application functions must be threadsafe. If the functions are re-entrant, performance will not decrease.
5. Do not forget to set the Content-Type header if you are using HTTPD_write() to send data back to the client.
6. Scour your code for memory leaks.
7. Think about your error paths. If you generate error messages yourself and send them back as HTML, you should return HTTPD_OK from your service function(s).

GWAPI reference information

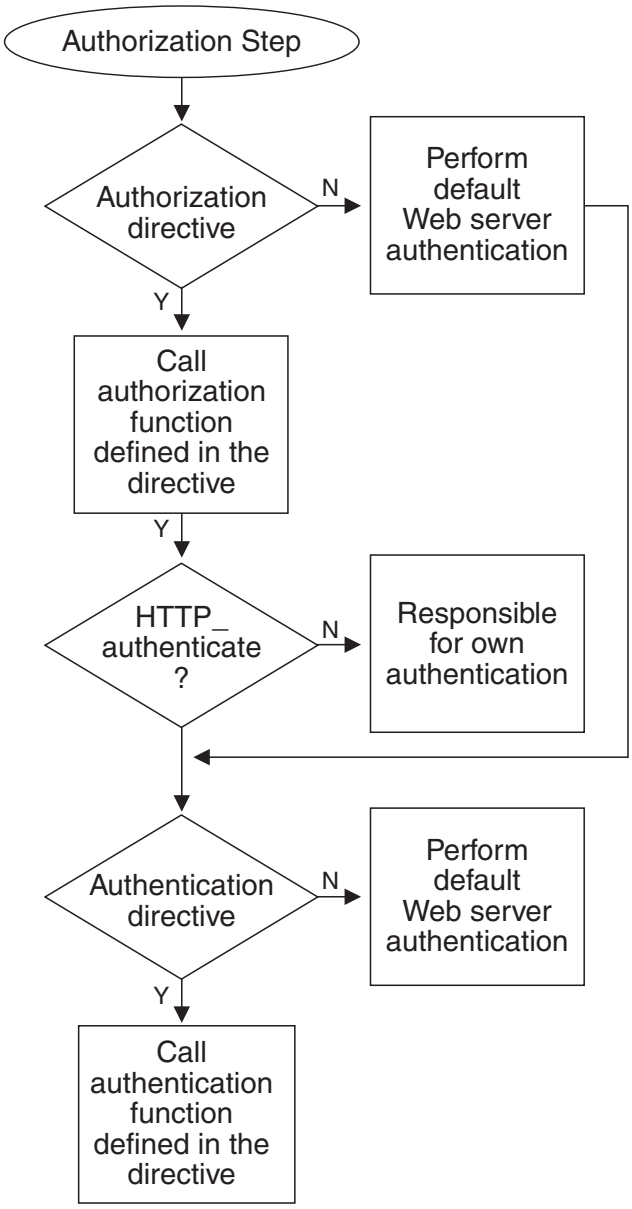
Authentication and authorization

First, a short review of the terminology:

Authentication
: The verification of the security tokens associated with this request

Authorization
: Process using the security tokens to determine if the requester has access to the resource

A visual overview of the server's authentication and authorization process:



In the HTTP Server, authentication is part of the authorization process; it occurs only when authorization is required.

If your GWAPI application provides its own authorization process, it will override the default server authorization and authentication. Therefore, if you have Authorization directives in your configuration file, the application functions associated with them must also handle any necessary authentication. The predefined HTTPD_authenticate() function is to assist you with providing authentication.

There are three ways you can provide for authentication in your authorization application functions:

- Write your own separate authorization and authentication application functions. In your configuration file, use both the Authorization and the Authentication directives to specify these functions. Be sure to include HTTPD_authenticate() in your authorization application function.

When the Authorization step is executed, it will perform your authorization application function which will, in turn, call your authentication application function.

- Write your own authorization application function but have it call the default server authentication. In your configuration file, use the Authorization directive to specify your function. In this case, you will not need the Authentication directive. Be sure to include HTTPD_authenticate() in your authorization application function.

  When the Authorization step is executed, it will perform your authorization application function which will, in turn, call the default server authentication.
- Write your own authorization application function and include all your authentication processing right in it. Do not use HTTPD_authenticate() in your authorization application function. In your configuration file, use the Authorization directive to specify your function. In this case, you will not need the Authentication directive.

  When the Authorization step is executed, it will perform your authorization application function and any authentication it included.

If your GWAPI application does not provide its own authorization process, you can still provide customized authentication. To do this:

- Write your own authentication application function. In your configuration file, use the Authentication directives to specify your function. In this case, you will not need the Authorization directive.

When the Authorization step is executed, it will perform the default server authorization which will, in turn, call your authentication application function.

Things to remember:

If you do not have any Authorization directives in your configuration file, or their specified application functions decline to handle the request, the server's default authorization will occur.

If you do have Authorization directives in your configuration file and their application functions include HTTPD_authenticate(), the server will call any authentication functions specified in the Authentication directives. If you do not have any Authentication directives defined, or their specified application functions decline to handle the request, the server's default authentication will occur.

If you do have Authorization directives in your configuration file but their application functions do not include HTTPD_authenticate(), no authentication functions will be called by the server. You must code your own authentication processing as part of your authorization application functions or make your own calls to other authentication modules.

The HTTP Server will automatically generate the challenge (prompting the browser to return userID and password) if you return 401 or 407 from your authorization exit. However, you must still configure a protection setup so that this will occur correctly.

Environment variables

For a description of variable names you can use in the Web server predefined functions, HTTPD_extract() and HTTPD_set(), see Appendix D, "Environment

variables" on page 583. These variables contain values you can extract from a client request (read-only) or values you can set or create when processing a client request.

Notes:

1. User-defined variable names cannot have a prefix of **SERVER_**. The GWAPI function reserves any variable starting with **SERVER_** for the server and, therefore, is READ ONLY.
2. All headers sent by the client (such as Set-Cookie) are prefixed by "HTTP_" and their values can be extracted. To access variables that are headers, prefix the variable name with "HTTP_". You can also create new variables using the HTTPD_set() predefined function. See RCF 2068 for details about these headers.

HTCodePage_t, the other data type

The HTCodePage_t enumeration data type is defined in the HTAPI.h file to support specification of a code page by GWAPI users:

```
typedef enum _HTCodePage_t
{
   BINARY = 0,
   ASCII,
   EBCDIC,
   RAWNET,
   RAWFILE
}
   HTCodePage_t;
```

type	conversion required
BINARY	no conversion
ASCII	equivalent to the code page specified by the DefaultNetCp configuration directive Default value: `ISO8859-1`
EBCDIC	equivalent to the code page specified by the DefaultFsCp configuration directive Default value: `IBM-1047`
RAWNET	data from network socket, format known to converter
RAWFILE	data from host file, format known to converter

An example of this data type declaration is:

```
HTCodePage_t codepage;
```

z/OS GWAPI REXX applications

This section describes support for writing z/OS GWAPI exit routines in REXX. The GWAPI REXX application supports two major activities:

- Invoking REXX executable programs from GWAPI exit routines

  GWAPI REXX support allows REXX executable programs to be plugged in at the following GWAPI exit routines:

  – Authentication
  – Authorization
  – DataFilter

- Error
- Log
- NameTrans
- Object Type
- PICSDBLookup
- PostExit
- PreExit
- ServerInit
- ServerTerm
- Service

Note: When calling the above exit routines, the first parameter MUST be the handle parameter

- Accessing GWAPI functions from REXX executable programs

  Standard GWAPI support provides a set of functions for use by exit routines. As part of the GWAPI REXX support, a corresponding set of linkage routines are provided that enable REXX executable programs to call the GWAPI functions. The following table lists the REXX and GWAPI function names and a function description:

REXX Function Name	GWAPI Function Name	Function Description
IMWXATR	HTTPD_attributes	Extracting file attributes
IMWXAUT	HTTPD_authenticate	Invoking the authentication function
IMWXFIL	HTTPD_file	Sending a file to the client
IMWXLGA	HTTPD_log_access	Writing to the server access log
IMWXLGT	HTTPD_log_trace	Writing to the server trace log
IMWXLOG	HTTPD_log_error	Writing to the server error log
IMWXPXY	HTTPD_proxy	Making a proxy request
IMWXRD	HTTPD_read	Reading input from the client
IMWXRVT	HTTPD_reverse_translate	Reverse translating a file system path to URL
IMWXSET	HTTPD_set	Setting server and local user variables
IMWXSLB	HTTPD_supply_label	Supplying a PICS label
IMWXTRN	HTTPD_translate	Translating a URL to a file system path
IMWXWRT	HTTPD_write	Writing output to the client
IMWXXTR	HTTPD_extract	Extracting server and local variables

Invoking REXX executable programs as GWAPI applications

Before invoking a REXX executable program as a GWAPI application, you need to know about:

- GWAPI REXX DLL and function names

  GWAPI REXX processing begins when a GWAPI directive is matched and the directive specifies one of the GWAPI REXX function names. The GWAPI REXX

function establishes a REXX environment, then executes the REXX executable program whose name it derived from information on the directive and in the URL.

The GWAPI REXX DLL and function names are:

ServerInit exit
IMWX00.so:IMWXSI

DataFilter exits for open, write, and close, respectively
IMWX00.so:IMWXD1:IMWXD2:IMWXD3

All other GWAPI exits
IMWX00.so:IMWX00

More than one GWAPI REXX function is necessary to accommodate variations in the arguments passed.

- REXX Exec Name and PATH_INFO

  When a GWAPI REXX function is invoked, the name of the REXX executable program is obtained from the server PATH_INFO variable. As it does for any GWAPI exit routine, the server creates PATH_INFO from information specified in the directive and the request URL. The following example illustrates the process:

  If the Service directive is defined as:

  ```
  Service /GWAPIRX* usr/lpp/internet/bin/IMWX00.so:
  IMWX00/usr/lpp/internet/server_root/cgi-bin*/Omega
  ```

  Note: The directive must be typed on one line, even thought it is shown here on two lines.

  If the request URL is:

  ```
  /GWAPIRX/userpgm.rx/Alpha/Beta
  ```

  The server assigns the following value to the PATH_INFO variable and passes it to the GWAPI REXX function:

  ```
  usr/lpp/internet/server_root/cgi-bin/userpgm.rx/Alpha/Beta/Omega
  ```

  The GWAPI REXX executable program expects the PATH_INFO string passed from the server to begin with the fully-qualified file name of the REXX executable program to be executed. Qualifiers are checked from left to right until a valid file name is found. Then, before invoking the executable program, GWAPI REXX redefines PATH_INFO by stripping the file name from the string. Assuming that GWAPI REXX finds the following valid file name:

  ```
  /usr/lpp/internet/server_root/cgi-bin/userpgm.rx
  ```

  PATH_INFO will be redefined to contain the remaining fragment:

  ```
  /Alpha/Beta/Omega
  ```

- Specifying directives for GWAPI REXX

  The following examples illustrate how to invoke a REXX executable program for each of the supported directives. The GWAPI REXX DLL symbolic link (IMWX00.so) default directory path is **/usr/lpp/internet/bin** and the GWAPI REXX executable program default directory path is **/usr/lpp/internet/server_root/cgi-bin**.

  EXAMPLE

```
ServerInit                /usr/lpp/internet/bin/IMWX00.so:
IMWXSI/usr/lpp/internet/server_root/cgi-bin/DGWInit.rx
Authentication Basic      /usr/lpp/internet/bin/IMWX00.so:
IMWX00/usr/lpp/internet/server_root/cgi-bin/Authn.rx
Authorization  /DB2*      /usr/lpp/internet/bin/IMWX00.so:
IMWX00/usr/lpp/internet/server_root/cgi-bin/Authz.rx
DataFilter                /usr/lpp/internet/bin/IMWX00.so:
IMWXD1:IMWXD2:IMWXD3/usr/lpp/server_root/cgi-bin/DataF.rx
Error          /*         /usr/lpp/internet/bin/IMWX00.so:
IMWX00/usr/lpp/internet/server_root/cgi-bin/Error.rx
Log            /cash*     /usr/lpp/internet/bin/IMWX00.so:
IMWX00/usr/lpp/internet/server_root/cgi-bin/Log.rx
NameTrans     /file*      /usr/lpp/internet/bin/IMWX00.so:
IMWX00/usr/lpp/internet/server_root/cgi-bin/PostEx.rx
ObjectType    /html       /usr/lpp/internet/bin/IMWX00.so:
IMWX00/usr/lpp/internet/server_root/cgi-bin/ObjTyp.rx
Post/Exit                 /usr/lpp/internet/bin/IMWX00.so:
IMWX00/usr/lpp/internet/server_root/cgi-bin/PostEx.rx
PreExit                   /usr/lpp/internet/bin/IMWX00.so:
IMWX00/usr/lpp/internet/server_root/cgi-bin/PreEx.rx
PICSDBLookup              /usr/lpp/internet/bin/IMWX00.so:
IMWX00/usr/lpp/internet/server_root/cgi-bin/PICSDBL.rx
ServerTerm                /usr/lpp/internet/bin/IMWX00.so:
IMWX00/usr/lpp/internet/server_root/cgi-bin/ICSTerm.rx
Service       /GWAPIRX*   /usr/lpp/internet/bin/IMWX00.so:
IMWX00*
```

Note: The directives must be typed on one line, even though it is shown here on two lines.

For descriptions of these directives, see Appendix B, "Configuration directives" on page 417.

- Errors invoking a GWAPI REXX executable program

  When a problem occurs attempting to invoke an executable program, GWAPI REXX writes the following message to the server error log:

  `Error` *`error_code`* `attempting to execute` *`exec_name`*

  error_code is one of the following values:

  –99 A storage request failed.

  –98 Necessary information, such as a path name or exec name, could not be extracted from the server variables.

  –97 Problems occurred processing the executable program file. Possible problems include non-existent file, improper permission, or no free file descriptor.

  –96 The executable program contained more than 16,384 lines.

  –95 BPXWRBLD service which creates z/OS REXX environment failed.

Writing GWAPI REXX Executable Programs

This section describes how to write GWAPI REXX executable programs.

GWAPI REXX executable program conditions

By the time a GWAPI REXX executable takes control, some environmental setup has been done. As with REXX CGIs, an array of REXX variables with stem name __environment has been defined. The variable __environment.0 contains the count. Note that environment variables must be prefixed by two underscores.

These contain the server variables passed to the exit. Each variable contains a string like the following:

```
REQUEST_METHOD=POST
```

In addition to the environment variables, arguments may be passed to the executable program, depending on the GWAPI exit and GWAPI REXX function involved:

IMWX00
No arguments

IMWXSI
Major version number, Minor version number

IMWXD1
OPEN

IMWXD2
WRITE

IMWXD3
CLOSE

Note in the GWAPI REXX directives shown earlier, only one REXX executable program is specified on the DataFilter directive; the executable is called for all three DataFilter exits. The argument passed to the exec indicates whether the call is for OPEN, WRITE, or CLOSE.

Calling external subroutines and functions

Note the method of invoking the initial GWAPI REXX executable program differs from the technique used to invoke external subroutines or functions called from the initial exec. Calls from the initial exec to external subroutines or functions are handled as described in the *z/OS Using REXX to Access OpenEdition MVS Services* book. Basically, the unqualified executable program file names of external subroutines or functions are resolved using the list of HFS directories in the server PATH variable.

Debugging and problem determination

GWAPI REXX captures messages generated by the REXX interpreter during the execution of an executable program. This output, redirected to the server error log file, includes:

- Syntax and execution error messages
- Messages generated by TRACE and SAY instructions

Transience of REXX variables

REXX variables do not persist beyond the life of the GWAPI REXX exec that creates them. Therefore, it is not possible to pass information from one exit to another using REXX variables.

Calling GWAPI predefined functions

GWAPI applications must use GWAPI-defined services for certain functions, such as reading or writing client data, and manipulating server variables. The GWAPI REXX support provides linkage routines that enable execs to call these GWAPI-defined services. (Familiarity with the GWAPI HTTPD_xxxx service is assumed.)

To invoke a GWAPI service, an exec issues an ADDRESS LINKMVS command to call its corresponding link routine:

```
ADDRESS LINKMVS 'IMWXxxx  parm1  parm2   ...  parmj'
```

The link routines provided in the GWAPI REXX support are described below. The following points apply to all of the GWAPI REXX link routines:

- The GWAPI handle is automatically supplied to the underlying service.
- Only the character data values are specified to the routines. The length parameters required by the underlying GWAPI services are derived from the lengths of the REXX variables. A variable used for output must be defined, and be of sufficient length if the output may exceed 500 bytes.
- The REXX variables are limited to a maximum length of 32,767 bytes by the LINKMVS command. If the return code from ADDRESS LINKMVS is -2, the REXX routine may have to be coded so that no variable exceeds 32,767 bytes (for example, loop on writes until all data is written).
- The return code value from the underlying GWAPI service is passed back to the exec in the REXX RC variable, with one exception. If a routine returning output finds that a supplied variable is too short to contain the output data, it usually returns a value of 6 (HTTPD_BUFFER_TOO_SMALL). Because of the way LINKMVS works, the output variable is shorter than the output data and the output data exceeds 500 bytes. In this event, the variable RC will be set to the length required to contain the output data. If there is more than one output field, RC will be set to the maximum of the output lengths. Basically, if RC is greater than 500, retry the function after redefining the output fields with lengths equal to RC. For detailed information about the LINKMVS command, see the *REXX/MVS Reference Guide*.

Link routines to GWAPI functions

The following describes each of the GWAPI REXX routines.

IMWXATR (calls HTTPD_attributes)

This routine is used to extract the attributes of a file. This example shows how to call the routine to retrieve file attributes:

```
filenm = '/tmp/usr/demofile.text'
fileattr = ' '
ADDRESS LINKMVS 'IMWXATR filenm fileattr'
```

IMWXAUT (calls HTTPD_authenticate)

This routine may be used by PreExit or Authorization execs to call Authentication. There are no parameters. The format is:

```
ADDRESS LINKMVS 'IMWXAUT'
```

IMWXFIL (calls HTTPD_file)

This routine is used to send a file to the browser from the server. This example shows how to send a file:

```
fname = '/u/gump/sample.txt'
ADDRESS LINKMVS 'IMWXFIL  fname'
```

IMWXLGA (calls HTTPD_log_access)

This routine is used to write a message to the server access log. This example shows how to call the routine:

```
msg = 'Application XYZ started for ICAPIRX/abc.rexxexec'
ADDRESS LINKMVS "IMWXLGA  msg'
```

IMWXLGT (calls HTTPD_log_trace)

This routine is used to write a message to the server trace log. This example shows how to call the routine:

```
msg = 'Application XYZ initializing...'
ADDRESS LINKMVS 'IMWXLGT  msg'
```

IMWXLOG (calls HTTPD_log_error)

This routine is used to write a message to the server error log. This example shows how to call the routine:

```
msg = 'Application XYZ could not find dataset A.B.C.'
ADDRESS LINKVMS 'IMWXLOG  msg'
```

IMWXPXY (calls HTTPD_proxy)

This routine is used to make a proxy request. This example shows the format:

```
ADDRESS LINKMVS 'IMWXPXY proxyurl reqbody'
```

The REXX variable proxyurl contains the proxy url and reqbody contains the body of the request.

IMWXRD (calls HTTPD_read)

This routine is used to read input from the requester. The following example shows how to ready the client input, assuming that the REXX variable cntntlen contains the length from CONTENT_LENGTH:

If the data is to be converted to EBCDIC, CONVERT_REQUEST_BODY should be set to YES.

If the data is to remain ASCII, CONVERT_REQUEST_BODY should be set to NO. This is the default.

```
inarea = substr(' ',1,cntntlen)
ADDRESS LINKMVS 'IMWXRD  inarea'
```

IMWXRVT (calls HTTPD_reverse_translate)

This routine is used to translate a file system path to a URL. This example assumes that the REXX variable fspath contains a file name, and url will receive the output URL.

```
ADDRESS LINKMVS 'IMWXRVT  fspath url'
```

IMWXSET (calls HTTPD_set)

This routine is used to set a server variable. The following example shows how to set variable MY_OWN_VARIABLE:

```
myname = 'MY_OWN_VARIABLE'
myvalue = 'whatever I want'
ADDRESS LINKMVS 'IMWXSET myname myvalue'
```

IMWXSLB (calls HTTPD_supply_label)

This routine is used to provide a PICS label. This example shows the format:

```
ADDRESS LINKMVS 'IMWXSLB  data'
```

The REXX variable data contains label data.

IMWXTRN (calls HTTPD_translate)

This routine is used to translate a URL to a file system path. The following example assumes that REXX variable url contains the input URL, and REXX variables trurl, pathtr, and qstr will receive the translated URL, PATH_TRANSLATED, and QUERY_STRING outputs, respectively:

```
ADDRESS LINKMVS 'IMWXTRN url trurl pathtr qstr'
```

IMWXWRT (calls HTTPD_write)

This routine is used to write output to the requestor. The following example shows how to write a few lines of output to the client browser in one call:

```
stuff = '<PRE>This is the first line<BR>'
stuff = stuff 'and this is the second line<BR>'
stuff = stuff 'and here is the last line</PRE><HR>'
ADDRESS LINKMVS 'IMWXWRT stuff'
```

Note that IMWXWRT does not append a newline to output, so line breaks must be explicitly specified by the application.

A GWAPI REXX executable program cannot use the **cgiutils** command to set output headers. Output headers must be set through the appropriate server variables (for example: CONTENT_TYPE, CONTENT_ENCODING) using IMWXSET.

IMWXXTR (calls HTTPD_extract)
This routine is used to retrieve the value of a server variable. The following example shows how to get the value of variable STATUS_INDICATOR:

```
statname = 'STATUS_INDICATOR'
statind = ' '
ADDRESS LINKMVS ' IMWXXTR statname statind'
```

GWAPI REXX executable program exit conditions:

When a GWAPI REXX executable program exits, it must set a numeric completion code with the EXIT instruction.

When the exec completion code is within the range of HTTP return codes (-1 to 599), the value is passed unchanged to the server, and the server disposes of the request accordingly. When the exec completion code is outside the HTTP range, GWAPI REXX assumes it is an error indication and writes the following message to the server error log:

```
Completion code from REXX exec exec_name is completion_code
```

A completion code of HTTP_BAD_REQUEST (400) is passed back to the server.

Example GWAPI REXX service executable program

The following REXX executable program, which runs out of the Service exit, performs the following functions:

- Sets ICS variables in preparation for output
- Displays the set of input environment variables
- Prints an HFS file
- Displays an image
- Sets appropriate return code.

Important syntax note: Environment variables must be prefixed by two underscores.

EXAMPLE

```
/* REXX */

/***      Set HTTP Server variables for output      ***/
name = 'CONTENT_TYPE'
value = 'text/html'
ADDRESS LINKMVS 'IMWXSET name value'
name = 'CONTENT_ENCODING'
value = 'ebcdic'
```

```
ADDRESS LINKMVS 'IMWXSET name value'

/***      List environment variables          ***/
output = 'List of all'  __environment.0 'Environment Variables:<PRE>'
do e = 1 to __environment.0
  output = output right(e,,'0')'.' __environment.e '<BR>'
end
output = output || '</PRE><HR>'
ADDRESS LINKMVS 'IMWXWRT output'

/***     Print a file        ***/
filename = '/etc/httpd.envvars'
output = 'Display HFS file' filename '<PRE>'
ADDRESS LINKMVS 'IMWXWRT output'
ADDRESS LINKMVS 'IMWXFIL filename'
hfsrc = rc
output = '</PRE>'
ADDRESS LINKMVS 'IMWXWRT output'
if hrsrc ¬=0 then output = 'IMWXFIL failed with rc:' hfsrc
output = output '<HR>'
ADDRESS LINKMVS 'IMWXWRT output'

/***       Display an image        ***/
out = 'Is this an image, or what <img src='/icons/unknown.gif">MHR>'
ADDRESS LINKMVS 'IMWXWRT out'

/***          Set successful return code       ***/
exit 200
```

GWAPI REXX downloads

To assist you in writing GWAPI REXX applications, a package of routines that can be called from REXX execs can be found at:

```
http://www.ibm.com/s390/nc/sntc/DB2IMSRX.html
```

DB2IMSRX contains routines for REXX execs (GWAPI or CGI). The routines are:

FORMWWWX
: Parses client form inputs (POST or GET) into REXX variables

DB2WWWX
: Executes an SQL statement or a DB2 stored procedure using REXX variables inputs or outputs

IMSWWWX
: Executes an IMS transaction (using APPC) using REXX variables as inputs and outputs

Example GWAPI REXX data filter executable program

The following REXX executable program, which runs out of the Data Filter exit, opens a data filter that will search the datastream looking for Kappa and convert the string to KAPPA.

EXAMPLE

```
/* REXX */

type = arg(1)
say 'Type is 'type
select
  when type = 'OPEN' then do;
    nm = 'DOCUMENT_URI'
    ADDRESS LINKMVS 'IMWXXTR nm vl'
```

```
    if rc = 0 & POS('rexx',vl) = 0 then orc = 000
    else orc = 200
    nm = 'DOCUMENT_URI'
    ADDRESS LINKMVS 'IMWXXTR nm vl'
    log = ''
    ADDRESS LINKMVS 'IMWXLOG log'
    exit orc
  end

  when type = 'WRITE' then do;
    indata = arg(2)
    outdata = ''
    do until indata = ''
      parse var indata piece 'Kappa' indata
      outdata = outdata || piece
      if indata ^= '' then outdata = outdata || 'KAPPA'
    end
    say 'OUTDATA IS: 'outdata
    ADDRESS LINKMVS 'IMWXWRT outdata'

    nm = 'DOCUMENT_URI'
    ADDRESS LINKMVS 'IMWXXTR nm vl'
    wrc = 200
    log = ''
    ADDRESS LINKMVS 'IMWXLOG log'
    exit wrc
  end

  when type = 'CLOSE' then do
    nm = 'DOCUMENT_URI'
    ADDRESS LINKMVS 'IMWXXTR nm vl'
    crc = 200
    log = ''
    ADDRESS LINKMVS 'IMWXLOG log'
    exit crc
  end

  otherwise exit 400

end
```

Debugging C/C++ GWAPI programs

You can use the z/OS Debug Tool to debug your C/C++ GWAPI programs. This tool is shipped as part of the Language Environment (LE) component of z/OS.

Overview of support and restrictions

This interactive tool allows you to debug C/C++ GWAPI programs remotely from your workstation. Set a break point with the simple click of the mouse. Use the windowing capabilities of your workstation to view multiple segments of your source and your storage, while monitoring a variable at the same time.

Note the following restrictions:

- The Debug Tool supports only one debugging operation on the Web server at a time.
- The Debug Tool supports multi-threaded requests. However, you cannot debug multiple requests concurrently. If the tool has started debugging a request, subsequent requests will be queued and started when the previous request completes.

- You cannot start the Debug Tool from the MVS console. After you set the required Web server directives, the tool will start automatically when the Web server is started.
- To configure the Web server, you must manually update the configuration file; there is currently no support for using the Configuration and Administration Forms graphical user interface.

Getting started

This quick start example shows the required steps for enabling and using the Debug Tool:

Step 1. Install the remote debugger

The remote debugger used with the z/OS Debug Tool is the z/OS C/C++ Productivity Tools Debugger. This debugger is the workstation component and graphical user interface for the Debug Tool.

For more information, go to URL:

`http://www.ibm.com/software/ad/c390/pt/dbg1.htm`

For information on other C/C++ Productivity Tools, go to URL:

`http://www.ibm.com/software/ad/c390/pt/`

Step 2. Compile your C/C++ program with the TEST compile-time option

Before testing your C/C++ program with the Debug Tool, you must compile it with the TEST option.

For instructions, see the *Debug Tool User's Guide and Reference*.

To access Debug Tool documentation on the Web, go to URL:

`http://www.ibm.com/s390/dt/`

Step 3. Edit the Web server configuration file

To enable support for the Debug Tool, update the following directives in the configuration file:

- "DebugToolAddr - Identify the workstation running the Remote Debugger" on page 485
- GWAPI directives

  Each step in the request process has a configuration directive that allows you to indicate which of your application functions you want called and executed during that step.

  To enable the Debug Tool, you simply add the prefix **dbg** to the appropriate GWAPI directive. For example, to debug a GWAPI program that is called during the PreExit step, add the prefix **dbg** to the PreExit directive in the configuration file:

  dbgPreExit */path/file:function_name*

  For a list of GWAPI directives, see "GWAPI directives and syntax" on page 361.

Step 4. Start the remote debugger on the workstation

Step 5. Start or restart the Web server

Troubleshooting hints and tips

Debug Tool hints and tips

To access Debug Tool hints and tips on the Web, go to URL:

`http://www.ibm.com/s390/dt/`

Web server error messages

Messages IMW0346E-IMW0349E will be issued to the console or to the error log if there are configuration or other errors related to the Debug Tool.

You can use the -grep command to help locate an error message. The command syntax is:

-grep -*message_number*

If you cannot determine the cause of the problem from error messages, the -vv trace log can provide more information.

Chapter 20. Accessing LDAP information with the LDAP API

Overview of LDAP API 379
Accessing LDAP configuration information . . . 379
Sample LDAP program 380

Overview of LDAP API

The HTTP Server allows access to LDAP information through the use of a plug-in or API.

For example, your plug-in could open its own connection (using ldap_open) to the LDAP server that the HTTP Server is using to access shared information. The plug-in would need information about the LDAP server, such as host name or port numbers, to do this. The API could bind using ldap_bind to connect and authenticate with the server, then search using ldap_search for the relevant LDAP information.

LDAP uses a set of DAP functions which decreases your system requirements:

Table 5. LDAP Access to X.500

Function	Use
ldap_bind	connect and authenticate with an LDAP server
ldap_unbind	log off and disconnect from an LDAP server
ldap_abandon	disconnect from a non responding LDAP server
ldap_search	query an LDAP server
ldap_add	add a new entry
ldap_delete	delete an entry
ldap_modify	change an existing entry
ldap_rdn	get the relative distinguished name

For information on configuring your Web server to access LDAP servers, see Chapter 15, “Retrieving Lightweight Directory Access Protocol information” on page 297.

To view LDAP documentation, go to the z/OS Internet Library at URL:
`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

Accessing LDAP configuration information

To connect to the same LDAP server that the HTTP Server uses to store information, you must access the information in the server configuration file. This section describes how to extract server information for use in your own API.

You can access the LDAP configuration information using the HTTPD_extract call. Parameters to access the configuration file using HTTPD_extract are:

LDAP:LABELLIST
Retrieves all labels defined in LDAPInfo directives in the configuration file. Each label is separated by a new line character.

LDAP:LABEL:<label>:<fieldName>
retrieves the value of the subdirective for the corresponding label in the LDAPInfo directive. Any LDAPInfo subdirective is a value for *fieldName*. For example, to access the server authentication type of the LDAPInfo directive labeled `PrimaryLdapServer`, use the extract call:

```
HTTPD_extract (handle,"LDAP:LABEL:PrimaryLdapServer:ServerAuthType", ...)
```

fieldName above has an additional value not included in the LDAPInfo subdirectives. ServerPassword is a valid field name, and returns the decrypted password from the stash file for LDAP authentication.

LDAP:LABEL:CURRENT:<fieldName>
Uses the currently used LDAPInfo directive to retrieve the value of the specified subdirective. An LDAPInfo directive is *current* when it is associated with a protection setup that has matched the current request.

fieldName above has an additional value not included in the LDAPInfo subdirectives.ServerPassword is a valid field name, and returns the decrypted password from the stash file for LDAP authentication.

Note: For more information, see the description of the LDAPInfo directive in "LDAPInfo - Define an external LDAP server" on page 485.

After you access the LDAP server information from the configuration file, a log file named LDAP_TRACE_LOG is created in the LOGS directory . The log file reports successful and unsuccessful readings from the configuration file.

Sample LDAP program

You can link to a sample LDAP API program on the Web site at URL:
`http://www.ibm.com/software/webservers/httpservers/doc53.html`

The example code obtains the labels, or subdirective values, in all `LDAPInfo` directives in the configuration file. A log file named ldap is created containing information on the labels attempted and read. It will also display the values for each label obtained successfully.

Part 7. Appendixes

Appendix A. Commands

cacheagt command 383
Syntax. 383
Flags 384
Example 384
cgiparse command 384
Syntax. 384
Flags 386
Examples. 386
Exit statuses. 387
cgiutils command 387
Syntax. 387
Flags 387
Examples. 389
htadm command 389
Syntax. 389
Flags 389
Examples. 392
htlogrep command 392
Syntax. 392
Flags 392
Examples. 393
Standalone mode 393
Scalable Server mode. 393
httpd command 393
Syntax. 393
Flags 394
Tracing flags. 396
Module names 398
Signal handling. 399
Examples. 399
IMWHTTPD program 400
Syntax. 400
Parameters 402
LE run-time option 402
Setting server environment variables . . . 404
Tracing options. 405
Signal handling. 408
Examples. 409
IMWIWM PROC (workload management). . . 409
Parameters 409
wwwcmd command 409
Syntax. 410
Parameters 410
Action. 410
z/OS MODIFY console command 410
Syntax. 410
Parameters 411
Tracing parameters 411
Module names 413
Examples. 414
z/OS Workload Management console commands 415
WLM mode 415
Application environments 415
WLM systems 416

cacheagt command

The Web server proxy caching program is CACHEAGT. This program is located in the *install-path*/bin directory. The Web server automatically runs the CACHEAGT program at midnight if you enable proxy capability and you set the AutoCacheRefresh directive to **On** in the `javelin.conf` file. The CACHEAGT program reads the CacheAccessLog file to refresh the contents of the proxy cache. You need to understand the CACHEAGT program requirements to run the CACHEAGT command. For information on the CACHEAGT program, including input requirements, see "Understanding the CACHEAGT program" on page 307.

The CACHEAGT command enables you to execute the CACHEAGT program from a command line.

Before you use the CACHEAGT command, make sure that you have the same PATH, LIBPATH, and NLSPATH in your z/OS UNIX System Services shell environment that your Web server uses. For example, set the _CEE_ENVFILE environment variable to `/etc/httpd.envvars`.

Syntax

```
cacheagt [-vv] -r /path/javelin.conf
```

Flags

-vv
Turns on tracing. Put the flag in front of the -r flag so that any errors associated with the javelin configuration file are traced. You can write your trace messages to a file by using the tracelog directive. For information on the directive, see "Understanding the CACHEAGT program" on page 307.

-r */path*/**javelin.conf**
Specify the path and name of the proxy server configuration file. The proxy server requires the file name, `javelin.conf`.

Example

```
cacheagt -vv -r /etc/javelin.conf
```

cgiparse command

Use the cgiparse command to parse the QUERY_STRING environment variable (in the case of GET method) or standard input (in the case of POST method) for CGI scripts. The cgiparse command can also be used to read CONTENT_LENGTH characters from standard input. All returned output is written to its standard output.

The QUERY_STRING is a string of EBCDIC text. Certain characters (+, <, >, (,), %, ″, /, ?, and space) are escaped by the client. Spaces are replaced by a plus sign (+), therefore, a real plus sign must be encoded. The other special characters are set to a percent character followed by the hexadecimal form of the character in the ASCII (codepage ISO8859-1) character set. The following examples show the hexadecimal form for each special character:

```
+       %2B
?       %3F
space   +
<       %3C
>       %3E
(       %28
)       %29
%       %25
"       %22
/       %2F
```

Syntax

```
cgiparse -Flag [Modifier]
```

Flags

Flags have one-character equivalents: -k -f -v -r -i -s -p -c -q -N -F -P

-keywords
Parses QUERY_STRING for keywords. Keywords are decoded and written to standard output, one per line. A space (+) is interpreted as the delimiter between keywords.

-form
Parses QUERY_STRING as form request. Returns a string which, when evaluated by the shell, will set shell variables with the prefix FORM_ followed by a field name. Field values are the contents of the variables. Variable names on a form request are restricted to include only alphabetical or numeric characters that are in the single byte character set.

-value *field-name*
Parses QUERY_STRING as form request. Returns only the value of *field-name*.

-read
Reads CONTENT_LENGTH characters from standard input and writes them to standard output.

-init
If QUERY_STRING is not set, reads the value of standard input and returns a SET statement that sets QUERY_STRING to this value. This can be used with both the GET and POST methods. A typical use is:

```
eval 'cgiparse -init'
```

This will set the QUERY_STRING environment variable, regardless of whether the GET or POST method was used.

cgiparse may be called multiple times in the same script when the GET method is used, but it should only be called once if the POST method is used. With the POST method, after standard input is read, the next cgiparse would find standard input empty and would hang.

-sep *separator*
Specifies the string used to separate multiple values. If you are using the **-value** flag, the default separator is newline. If you are using the **-form** flag, the default separator is a comma (,).

-prefix *prefix*
Used with **-POST** and **-form**, specifies the prefix to use when creating environment variable names. The default is ″FORM_″ if prefix is not specified for **-POST** and **-form**.

-count
Used with **-keywords**, **-form**, and **-value**, returns a count of items related to these flags.

-keywords
Returns the number of keywords

-form
Returns the number of unique fields (multiple values are counted as one)

-value *field-name*
Returns the number of values for *field-name* (if there is not a field named *field-name*, output is 0).

-*number*
Used with **-keywords**, **-form**, and **-value**, returns the specified occurrence related to these flags.

-keywords
Returns the *n*'th keyword. (For example **-2 -keywords** outputs the second keyword.)

-form
Returns all the values of the *n*'th field. (For example **-2 -form** outputs all the values of the second field.)

-value *field-name*
Returns the *n*'th of the multiple values of field *field-name*. (For example **-2 -value -whatsit** outputs the second value of the **whatsit** field).

-quiet
Suppresses all error messages. (Non-zero exit status still indicates error.)

-fscp *FileCodepage*
The FileCodepage is the name of the file system codepage used in codepage conversion when processing text document bodies. When an unknown codepage is set, the default is used. The default is IBM-1047. Specifying **-fscp** on the httpd command overrides the default set in the server configuration file (httpd.conf).

-netcp *NetCodepage*
The NetCodepage is the network codepage name used in codepage conversion when processing text document bodies. When an unknown codepage is set, the default is used. The default is ISO8859-1 if the -netcp option is not specified. Specifying **-netcp** on the httpd command overrides the default set in the server configuration file (httpd.conf).

Note: The system iconv() service MUST support conversion between the pair of codepages specified as DefaultFsCp and DefaultNetCp.

Some acceptable values are:

netcp	**fscp**
ISO8859-1	IBM-1047
IBM-932C	IBM-939
IBM-eucJC	IBM-939

-POST
Information from stdin is directly decoded and parsed into shell variables, QUERY_STRING is not used. **-POST** is equivalent to consecutive use of the **-init** and **-form** options.

Examples

The following examples ignore the fact that, in reality, QUERY_STRING is already set by the server. In the following examples, $ is the Bourne shell prompt.

- Keyword Search

```
$ QUERY_STRING="is+2%2B2+really+four%3F"
$ export QUERY_STRING
$ cgiparse -keywords
is
2+2
really
four?
$
```

- Parsing All Form Fields

```
$ export QUERY_STRING="name1=Value1&name2=Value2%3f+That%27s+right%21";
$ cgiparse -form
FORM_name1='Value1'; FORM_name2='Value2? That's right!'
$ eval `cgiparse -form`
$ set | grep FORM
FORM_name1="Value1"
FORM_name2="Value2? That's right!"
$
```

- Extracting Only One Field Value

```
$ QUERY_STRING="name1=value1&name2=Second+value%3F+That%27s+right%21";
$ export QUERY_STRING
$ cgiparse -value name1
```

```
value1
$ cgiparse -value name2
Second value? That's right!
$
```

- Using a separator

```
$ export QUERY_STRING="name1=abc&name2=123&name1=def&name2=456";
$ cgiparse -form
FORM_name1='abc, def'; FORM_name2='123, 456'
$ cgiparse -form -sep /
FORM_name1='abc/def'; FORM_name2='123/456'
$
```

Exit statuses

0 Success

1 Illegal command line

2 Environment variables not set correctly

3 Failed to get requested information (for example, there is no such field or QUERY_STRING contains keywords when form field values are requested).

Note: When you receive one of these error codes, you may receive additional informational messages, too. The message varies depending on the command issued.

cgiutils command

Use the cgiutils command in no-parse header programs to produce a full HTTP 1.0 response.

Note: If you want to supply your own no-parse header (nph) programs specifically to return your own return values, the name of the program must begin with **nph-**. This prevents the server header from overriding your return value with the standard server return value.

Syntax

```
cgiutils -Flag [Modifier]
```

If *Modifier* contains blanks, enclose it in quotes.

Flags

-version
Returns version information.

-nodate
Does not return the Date: header.

-noel
Does not return a blank line after headers. This is useful if you want other MIME headers after the initial header lines.

-status *nnn*
Returns full HTTP response with status code *nnn*, instead of only a set of HTTP headers. Do not use this flag if you only want the Expires: header.

-reason *explanation*
Specifies the reason line for the HTTP response. You can only use this flag with the **-status** *nnn* flag.

-ct [*type/subtype*]
Specifies MIME Content-Type header. This example specifies a MIME content type of text/html:

```
cgiutils -ct text/html
```

If you omit the *type/subtype*, the MIME content type is set to the default text/plain. This example sets the MIME content type to text/plain.

```
cgiutils -ct
```

Specifies `text/x-ssi-html` to cause server side include processing of your output stream.

-ce *encoding*
Specifies MIME Content-Encoding header. For example:

```
cgiutils -ce  x-compress
```

Specifies ebcdic to force translation. All other codings prevent translation.

Note: Text encodings are expected in EBCDIC codepage IBM-1047 and will be translated to ASCII codepage ISO8859-1 by the server. If encoding is not explicitly specified, translation occurs if MIME type is not specified or if *type* is text.

-cl *language-code*
Specifies MIME Content-Language header. For example:

```
cgiutils -cl en_UK
```

-length *nnn*
Specifies MIME Content-Length header.

-expires *Time-Spec*
Specifies MIME Expires header. This flag specifies the time to live (the expiration date of a document) in any combination of days, hours, minutes and seconds. This is the length of time a document is considered valid. For example:

```
cgiutils -expires "2 days 12 hours"
```

The cgiutils command adds the time you specify to the current Greenwich Mean Time to determine the expiration date. The expiration date is put in the Expires: header in the HTTP format.

Note: If more than one word is used for Time-Spec, you must enclose the whole time specifier in double quotes.

-expires now
Produces an Expires: header that matches the Date: header. You should use this to prevent caching of your output.

-uri *URI*
Specifies the Universal Resource Identifier (URI) for the returned document.

-extra *xxx: yyy*
Specifies an extra header that cannot otherwise be specified for cgiutils.

Examples

- This example calculates the expiration date for the Expires: header.

  ```
  cgiutils -expires "1 year 3 months 2 weeks 4 days 12 hours 30 mins"
  ```

- The following example specifies a status code and reason, and sets the Expire: header equal to the Date: header.

  ```
  cgiutils -status 200 -reason "Virtual doc follows" -expires now
  ```

  This might produce headers similar to these:

  ```
  HTTP/1.0 200 Virtual doc follows
  MIME-Version: 1.0
  Date: Tue, 05 Jan 1996 03:43:46 GMT
  Expires: Tue, 05 Jan 1996 03:43:46 GMT
  ```

  The cgiutils command automatically produces the Server: header because it is available in the CGI environment. The Date: field is also automatically generated unless the -nodate flag is specified.

  This would include a blank line after the output to mark the end of the MIME header section. If you want to follow this with some more headers yourself, use the **-noel** (NO-Empty-Line) flag as shown in the next example.

- If you do not want the blank line after the header line, use the **-noel** flag:

  ```
  cgiutils -noel -expires "2 days" -nodate
  Expires: Tue, 07 Jan 1996 03:43:46 GMT
  ```

htadm command

Use the htadm command to control your IMWEBSRV server password files. Your server can use password files to control access to your files. See “Security concepts” on page 55 for information about controlling access to your server’s resources. With htadm you can add a user name to a password file, delete a user from a password file, change a password for a user, verify a user’s password, and create an empty password file. This command cannot be used to manage SAF userids and passwords.

Note: When you are running htadm as a separate line command and you are outside the server, you must ensure that the NLSPATH run-time environment variable is set. The NLSPATH variable should be set to:

```
NLSPATH=/usr/lpp/internet/%L/%N
```

Also, the following parameter should be set:

```
LANG=C
LANG=Ja_JP
```

Syntax

```
htadm -Flag  [Modifier]
```

Flags

-adduser *password-file* [*user-name* [**“* ”** | *password* *"real-name"*]]

Adds a user and password into the password file. If you enter the command with only *password-file*, you are prompted for the other parameters.

password-file
The path and name of the password file to which you want to add the user.

user-name
The name of the user you want to add.

Do not use the characters $, &, ^, in the user name, because these characters are interpreted by the shells first. At present, the C, Bourne, and K shells exhibit tendencies to truncate the data or interpret the data incorrectly.

The command fails if there is already a user of the same name in the password file.

password | **"* "**
The password you want to define for the user you are adding, or the asterisk character ("* "). The asterisk allows you to enter information for *real-name* as part of the command but still be prompted for the password.

Note: Quotes are required when specifying the asterisk character. If you omit the quotes, z/OS UNIX System Services will interpret the asterisk as a special z/OS UNIX character.

Passwords can be up to 32 characters long. Use only alphabetic and numeric characters for the password; do not use special characters.

Note: Some browsers are unable to read and send passwords longer than eight characters. Because of this limitation, if you define a password longer than eight characters, the server recognizes either the complete password or just the first eight characters of the password as valid.

"*real-name***"**
A comment or name you want to use to identify the user name you are adding. Whatever you enter will be written into the password file.

Note: Quotes are required if the *real-name* contains more than one word, for example, "Clark Kent".

-deluser *password-file* **[***user-name***]**
Deletes a user from the password file. If you enter the command with only *password-file*, you are prompted for the *user-name* parameter.

password-file
The path and name of the password file from which you want to delete a user.

user-name
The name of the user you want to delete. The command fails if the user name you specify does not exist in the password file.

-passwd *password-file* **[***user-name* **[***password***]]**
Changes the password for a user name already defined in the password file. If you enter the command with only *password-file*, you are prompted for the other parameters.

password-file
The path and name of the password file that contains the user name whose password you want to change.

user-name
The user name whose password you want to change. The command fails if the user name you specify does not exist in the password file.

password
The new password you want to define for the user name.

Passwords can be up to 32 characters long. Use only alphabetic and numeric characters for the password; do not use special characters.

Note: Some browsers are unable to read and send passwords longer than eight characters. Because of this limitation, if you define a password longer than eight characters, the server recognizes either the complete password or just the first eight characters of the password as valid.

-check *password-file* **[***user-name* **[***password***]]**
Verifies the password for a user name already defined in the password file and lets you know if it is correct or not. If you enter the command with only *password-file* you are prompted for the other parameters.

password-file
The path and name of the password file that contains the user name whose password you want to verify.

user-name
The user name whose password you want to verify. The command fails if the user name you specify does not exist in the password file.

password
The password that you want to verify. If the password you enter is the one defined for the user name, the command writes `Correct` to standard output and completes with a 0 return code. If the password you enter is not the one defined for the user name, the command writes `Incorrect` to standard output.

-create *password-file*
Create an empty password file.

password-file
The path and name of the password file that you want to create.

-stash *filename password*

Create or change the password stash file. This file contains the encrypted password the server uses to log into an LDAP server when reading access control lists or configuration directives. When accessing the LDAP server, the HTTP Server will log in using its Distinguished Name (from the ServerDN subdirective of the LDAPInfo directive) and the decoded password from the stash file.

Attention: Although the file is encrypted, use file system security measures to limit read/write access to this stash file.

filename
The directory and name of the stashed password file

password-file
The password used to access the LDAP server. This password will be encrypted within the file.

Examples

- To add a user to a password file:

```
htadm -adduser /usr/lpp/internet/server_root/heroes.pwd
 clark superman "Clark Kent"
```

  Note: The command appears on two lines for printing purposes. When you issue the **htadm** line command, it appears on one line.

- To delete a user from a password file:

```
htadm -deluser /usr/lpp/internet/server_root/heroes.pwd clark
```

- To create a stashed password file for LDAP access:

```
htadm -stash ldap_server_pwd secretpassword
```

  The example creates a file named ldap_server_pwd containing the encrypted string that represents the password secretpassword.

htlogrep command

The Web server logging and reporting program is HTLOGREP. This program is located in the /usr/lpp/internet/sbin directory. By default, the Web server automatically runs the HTLOGREP program at midnight to archive the logs and reports generated on the previous day and to create new files for the current day.

This command enables you to execute the HTLOGREP utility from a command line. For information on the configuration directives you use to set up and customize logging and reporting, see "Logging and Reporting - Customize logs and generate reports" on page 491. For information on Web server logging and reporting options, see Chapter 11, "Customizing logs and reports" on page 189.

Before using the htlogrep command, ensure that the _CEE_ENVFILE environment variable is set to:

install-path/etc/httpd.envvars

Syntax

```
htlogrep [-Flag [-Flag [-Flag..]]]
```

Flags

-c *path/configuration-file*
Specify the path and name of the Web server configuration file. If a path and name are not specified, the Web server uses the default path and configuration file, *install-path*/etc/httpd.conf.

-l *path/log-file*
Specify the path and name of the log file to be processed. The default path is the root directory of the Access Log and the date of the previous day. The Access Log root directory is specified on the AccessLog directive in the configuration file specified on the -c flag. If a log file is not specified, the Web server processes all the logs that have been configured in the configuration file.

-d *path/debug-file*
Specify the path and name of the Web server debug file. This file contains information on processing that took place during the execution of the HTLOGREP program. The default is *install-path*/server_root/reports/debug.out.

-s *WLM-subsystem-name*
This flag is required and valid when the Web server is running in Scalable Server mode. Specify the WLM subsystem name of the Web server for which reports are to be generated.

-w *number-of-minutes*
This flag is valid when the Web server is running in Scalable Server mode. Specify the number of minutes that the Web server will wait to execute the HTLOGREP program. This allows the Queue Servers time to close their log files when HTLOGREP is invoked at midnight. For immediate execution, set the value to zero (0). The default is 5 minutes.

Examples

In the following examples, the command is entered on a continuous line. For this book, some examples are split because of space restrictions.

Standalone mode

- To process logs for May 15, 2000, enter:

```
htlogrep -c install-path/etc/myhttpd.conf -l /mylogdir/httpd-log.May152000
```

- To process logs for May 15, 2000 with debug information going to a temporary file, enter:

```
htlogrep -c install-path/etc/myhttpd.conf -l /mylogdir/httpd-log.May152000
-d /tmp/myinfo.txt
```

Scalable Server mode

- To process logs for May 15, 2000, enter:

```
htlogrep -c install-path/etc/myhttpd.conf
-l /mylogdir/httpd-log.May152000.MYSUBSYSTEM
-s MYSUBSYSTEM
```

- To process logs for May 15, 2000 with debug information going to a temporary file, enter:

```
htlogrep -c install-path/etc/myhttpd.conf
-l /mylogdir/httpd-log.May152000.MYSUBSYSTEM
-d /tmp/myinfo.txt -s MYSUBSYSTEM
```

- To process logs for May 15, 2000 with no delay after the HTLOGREP program is invoked:

```
htlogrep -c install-path/etc/myhttpd.conf
-l /mylogdir/httpd-log.May152000.MYSUBSYSTEM
-w 0 -s MYSUBSYSTEM
```

Note: In this example each htlogrep command appears on multiple lines for printing purposes.

httpd command

Use the httpd command to start the server.

You can set all these flags (except -r) using the directives in the server configuration file.

Syntax

```
httpd [-Flag [-Flag [-Flag..]]]
```

Flags

To turn tracing on or off, see "Tracing flags" on page 396.

-B *bounce*
Starts the Web server with the sockets setting SO_REUSEADDR on. This will avoid a potential delay while sockets used by a previous invocation of the Web server are closed. This is especially useful if you have automation software that recognizes the Web server termination and needs to start a replacement Web server as quickly as possible.

By default, bounce is set on. There is no option on the httpd command to override the default of bounce on. To set this flag off, you must set the Bounce directive off in the httpd.conf file and omit -B from the Web server startup command.

If bounce is set off, the Web server will retry the bind to its listen ports for up to two minutes to allow previously used sockets to close. If the Web server is started quickly after a previous Web server shutdown and the startup option -B is not used, the listen ports may not be available. This can result in a delay in the availability of the Web server.

-drainsoc
Causes the server to drain TCP/IP sockets of all input data before closing the socket. This means that all input data still outstanding on the socket after the server has processed all the data it expects will be read and discarded. If web server tracing of TCP/IP communication is also being performed, any extra data read will be traced. This can be especially helpful in diagnosing potential browser software problems.

-fscp *FileCodepage*
DefaultFsCp. Specifies the name of the file system codepage used in codepage conversion when processing text document bodies. When an unknown codepage is set, the default is used. The default is IBM-1047. Specifying **-fscp** on the httpd command overrides the default set in the server configuration file (httpd.conf).

-gc_only
Only does garbage collection and then exits. This flag is used only for caching proxy servers.

-h HostName. Host names or IP addresses can be used on the template.

-netcp *NetCodepage*
DefaultNetCp. Specifies the network codepage name used in codepage conversion when decodingtext bodies on the network. When an unknown codepage is set, the default is used. The default is ISO8859-1 if the -netcp option is not specified. Specifying **-netcp** on the httpd command overrides the default set in the server configuration file (httpd.conf).

-nobg
Runs the server as a foreground process. Do not run the server as a background process.

-nolastmod
LastModified headers are not added to CGI output if they are not already present. The LastModified header affects whether or not browsers will cache the page.

-normalmode [on | off]
Turns HTTP connections on or off. If both normalmode and sslmode are off, the Web server will start in normal mode.

-nosec
Specifes the flag to use to force a base server. Security loads are bypassed.

-nosmf
Turns SMF recording off.

-nosnmp
Turns SNMP support off.

-p *port-number*
Listens on this port number. The default port number is 80. This flag overrides the Port directive specified in the configuration file.

-r *configuration-file*
Start the server with the specified server configuration file. You must use this flag if you want to start the server with a server configuration file other than the default /etc/httpd.conf.

Note: If you specify the -r option and either the normalmode or sslmode option, you **must** specify normalmode or sslmode before -r.

-restart
Restarts a server that is currently running.

-smf
Turns SMF recording on.

-snmp
Turns SNMP support on.

-sr *server-root*
Specify the current working directory of the Web server. This value overrides the directory on the ServerRoot directive in the configuration file.

In the following example, two Web servers are started using the httpd command. Both servers use the same configuration file, myhttpd.conf. Logging and Reporting directives are specified relative to the server root in the configuration file, for example, `AccessLog httpd-log`. The default install path, `/usr/lpp/internet`, is used. The first Web server is started by entering:

```
httpd -r myhttpd.conf -sr server1root
```

This specifies that the AccessLog file for the first Web server will be `/usr/lpp/internet/server1root/httpd-log`.

The second Web server is started by entering:

```
httpd -r myhttpd.conf -sr server2root
```

This specifies that the AccessLog file for the second Web server will be `/usr/lpp/internet/server2root/httpd-log`.

-ss CONNECTION_POOL
Specify where to place a shared memory segment for a connection pool. Explicitly specify a starting point for a shared memory segment and the size of the segment. Add this command only under the direction of IBM support personnel. IBM recommends the use of defaults: either do not use this command at all or explicitly code the defaults. This command is not valid without parameters.

```
-ss CONNECTION_POOL virtual_address segment_size
```

virtual_address
Code 0.

segment_size
Default=20 MB.

Refer to Chapter 1, "Planning for installation" on page 3 for more information on z/OS UNIX System Services parameters that may need to be set, especially, the following:
- IPCSHMSPAGES
- IPCSHMMPAGES
- IPCSHMNSEGS

See *z/OS UNIX System Services Planning* for more information about establishing the appropriate segment size.

-sslmode [on|off]
For a secure server, turns on the SSL protocol.

-sslport [port]
For a secure server, sets the port used for the SSL protocol.

-version
Returns the version number of the httpd executable and then exits.

Tracing flags

The level of tracing provided is -v trace, first level; -vv trace, second level; -mtv, third level; and -debug for maximum tracing. To trace caching, use the -vc option.

Notes:

1. We recommend that you turn tracing on only if instructed to do so by IBM support personnel. IBM support personnel can give you guidance on the most appropriate trace for your problem.
2. To turn tracing off, use the -nodebug option.
3. The –v, –vv, and –mtv flags represent a progression of increasing trace verbosity. They do not need to be used in combination with each other. If more than one of these flags is set, the first flag encountered sets the initial verbosity. All other flags are ignored.
4. The –debug and –nodebug flags are read from left to right by the Web server. If either flag is set without any modules specified, it should be set before the flag with modules is specified. Otherwise, the effect of the flag without the modules will be ignored.

   Example: Set flags as follows:

   ```
   httpd -debug -nodebug stacks
   ```
5. The order of placement of the –debug and –nodebug flags in the command line is independent of the placement of the –v, –vv, and –mtv flags. The –v, –vv, and –mtv flags are processed before the –debug and –nodebug flags. So, the –debug and –nodebug flags will always override the tracing state set by –v, –vv, and –mtv.

-v Verbose. Turns on first level debugging messages.

Verbose tracing provides basic tracing information on all requests without significantly impacting Web server performance. The design of the fields supports automatic parsing of the information. The format of the trace follows:

Table 6. Verbose trace format

Characters	Field
1-39	Thread Identifier and Timestamp :

Table 6. Verbose trace format (continued)

40	Space
41	'V' (indicates that this is a first-level tracing line)
42	Space
43-57	Client IP Address
58	Space
59-63	Port
64	Space
65-68	Request Method
69	Space
70-79	Content Length
80	Space
81-83	Response Code
84	Space
85-281	Event Timestamps
282	Space
283-on	Request URI

The `Event Timestamps` are given in elapsed seconds since January 1, 1970. The Web server lists these timestamps as entry-exit pairs, except for the `Request Completion Timestamp`. The elapsed time spent in each module is determined by calculating the difference between the timestamps. The eleven timestamp fields follow:

```
Acceptwait entry/exit
SSL Handshake entry/exit
Request Parsing entry/exit
Service Exit entry/exit
CGI Exit entry/exit
Request Completion Timestamp
```

Example:

```
  9E81518 01/Jun/2001:10:12:26.293840 : V       9.3.7.141    80
 GET      21239 200 0000000000.000000 0000000000.000000 00000000
00.000000 0000000000.000000 0991404746.080941 0991404746.081107
0000000000.000000 0000000000.000000 0000000000.000000 0000000000
.000000 0991404746.293835 /Admin/lgmast.gif
```

The example appears on a single line in the –v trace, even though it is shown here on multiple lines.

-vv
Very Verbose. Turns on second level debugging messages.

-mtv
Much Too Verbose. Turns on third level debugging messages.

-debug [*module_name*]
Turns on debugging for all modules or a specific module. **-debug** with no module name specified turns on debug for all modules.

-nodebug [*module_name*]
Turns off tracing for all modules or a specific module. **-nodebug** with no module name specified turns off debug for all modules.

-vc
Verbose cache. Turns on cache tracing messages.

Module names

On the -debug and -nodebug options, you can enter the following values for *module_name*:

bag
bags
cgi
config
connections
creators
dirbrw
dns
error
errors
fastcgi
format
frca
ftp
gc
gopher
groups
hash
hashtables
html
http
icapi
if
javelin
javelinbase
javelincfg
javelinpics
ldap
ldapv
lex
log
logging
mempool
mpool
mpoolv
netmonitor
nls
none
pics
proxy
proxycache
proxygroup
proxylock
proxylocks
request
resphdrs
ruledb
rules
selftest

ssi
stacks
status
stringlib
tcp
tcpip
threadinit
threadpool
time
timers
users
wbi
workqueue
workthread

Signal handling

The following sections explain how the server behaves when sent the SIGKILL, SIGTERM, and SIGHUP signals:

SIGKILL
This causes the server to terminate immediately. This can be accomplished by sending the SIGKILL signal to the httpd process using the z/OS UNIX shell command kill, the Web server wwwcmd command, or by cancelling the job using the z/OS operator console CANCEL command.

SIGTERM
This causes the server to stop and exit immediately. This can be accomplished by sending the SIGTERM signal to the httpd process using the z/OS UNIX shell command kill or the Web server wwwcmd command.

SIGHUP
Restart a running httpd. This causes httpd to stop accepting new requests, complete current requests, and, if there are no errors, reload the configuration file and resume processing. This can be accomplished by sending the SIGHUP signal to the httpd process using the z/OS UNIX shell command kill or the Web server wwwcmd command. If there are errors, you must fix the configuration file and try again.

The user ID that sends SIGTERM, SIGKILL, and SIGHUP signals to the httpd process must either be the user ID that started the Web server or a superuser ID. You need to know the Web server process ID in order to use the z/OS UNIX shell command, kill, or the Web server wwwcmd command to send the signal. The process ID is defined on the PidFile directive in the configuration file. For information about the kill command, see the *z/OS UNIX System Services Command Reference*.

Examples

- To start the server at port 8080, using the configuration file /usr/etc/httpd.conf instead of the default, /etc/httpd.conf, enter:

  ```
  httpd -p -8080 -r /usr/etc/httpd.conf
  ```

  If the Port directive is given in the configuration file, the **-p** flag is not required. However, the **-p** flag can be used to override the value set in the configuration file.
- To start the server using the default configuration file /etc/httpd.conf with verbose tracing:

```
httpd -vv
```

IMWHTTPD program

Use the IMWEBSRV PROC to start the server. The program is IMWHTTPD. IMWHTTPD requires either a rule file (default or -r) or a directory to export.

It is common practice to create a file named README containing instructions or notices to be read by anyone new to the directory. IMWHTTPD, by default, imbeds any README file in the hypertext version of a directory. The README file instructions can also be set with the DirReadme configuration directive.

Syntax

```
//IMWPROC PROC  LEPARM=,ICSPARM=
//*******************************************************************
//* PARM='LE runtime opts/ICS parms'
//*
//* LEPARM  ==> LE runtime opts
//* LEPARM='ENVAR("_CEE_ENVFILE=/etc/httpd.envvars.tmp")'
//*
//* ICSPARM ==> HTTP Server parameters
//* # Standalone HTTPD
//* ICSPARM='-p 8080
//* # WLM Queue Manager
//* ICSPARM='-SN WEBSN1 -p 8080
//* # WLM ApplEnv Queue Server
//* ICSPARM='-SN WEBSN1 -AE WEBHTML'
//*
//*  HTTP Server Parameters:
//*   -SN                         # WLM - subsystem name
//*   -AE                         # WLM - Application Envirnonment
//*
//*   -fscp      nnn              # File system codepage - EBCDIC
//*   -netcp     nnn              # net code page        - ASCII
//*
//*   -gc_only                    # clean cache & exit (garbage collect)
//*
//*   -normalmode
//*   -p         nnnn             # use port nnn (default 80)
//*   -sslmode
//*   -sslport   nnnn             # use port nnn (default 443)
//*   -nosec                      # no security
//*
//*   -nosmf                      # no smf processing on
//*   -smf                        # smf processing on
//*
//*   -r         /etc/httpd.conf  # use rule file xxxx
//*   -restart
//*   -v                          # trace to stderr
//*   -vv                         # trace to stderr
//*   -vc                         # cache trace to stderr
//*
//*   -version                    # show version and exit
//*
//*
//*
//*****************************************************************
//*
//*  The following LE runtime options are set as defaults within
```

```
//*  the main "C" module for the HTTP OS/390 Webserver.
//*
//*      POSIX(ON),ALL31(ON)
//*      ENVAR("_CEE_ENVFILE=/etc/httpd.envvars")
//*      ANYHEAP(4M,1M,ANY,FREE)
//*      BELOWHEAP(400K,50K,FREE)
//*      HEAP(4M,1M,ANY,KEEP,4K,4K)
//*      LIBS(1K,1K,FREE)
//*      STACK(200K,16K,ANY,FREE)
//*      HEAPP(ON)
//*      STORAGE(NONE,NONE,NONE,8192)
//*      TERMTHDACT(UADUMP)
//*      TRAP(ON)
//*      ABTERMENC(ABEND)
//*
//*  To verify the LE runtime options add the following line to
//*  the LEPARM symbolic definition for this proc.
//*
//*     RPTOPTS(ON)
//*
//*  The runopts report will be printed when the Webserver
//*  terminates.
//*-----------------------------------------------------
//*
//*  If extensive research is required to solve a problem the
//*  IBM customer service representative may ask that the
//*  problem be recreated and a SYSMDUMP furnished.
//*  If that is the case the, value of the RECOVERY
//*  configuration directive MUST be set to NONE.
//*  The SYSMDUMP DD statement listed below must be uncommented
//*  and completed.
//*  The filler <hlq> MUST be replaced with a valid high-level
//*  qualifier in the  DSNAME parameter.  This high-level
//*  qualifier must be valid in the customer environment.
//*  The recommended values for the SYSMDUMP DD statement
//*   have been provided.
//*  All values are changeable to fit the environment of the
//*  customer with the EXCEPTION of the DCB parameter.  The
//*  DCB parameter values are required for use with the IPCS
//*  dump reading tool and MUST NOT change.
//*
//*  The DCB parameter MUST have the following values:
//*          DCB=(LRECL=4160,BLKSIZE=24960,RECFM=FBS),
//*
//*  NOTE - The SYSMDUMP DD statement may be uncommented and
//*         completed at setup time.
//*
//*
//*******************************************************************
//WEBSRV   EXEC PGM=IMWHTTPD,REGION=0K,TIME=NOLIMIT,
//   PARM=('&LEPARM/&ICSPARM')
//*******************************************************************
//SYSIN    DD DUMMY
//OUTDSC   OUTPUT DEST=HOLD
//SYSPRINT DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//CEEDUMP  DD SYSOUT=*,OUTPUT=(*.OUTDSC)
```

```
//*SYSMDUMP DD DSNAME=<hlq>.WEBMDUMP.D&LYYMMDD..T&LHHMMSS,
//*          UNIT=SYSDA,
//*          SPACE=(CYL,(100,50)),
//*          DCB=(LRECL=4160,BLKSIZE=24960,RECFM=FBS),
//*          DISP=(NEW,DELETE,CATLG)
```

You should note that JCL only allows 100 characters in the PARM=' ' statement.

Parameters

LE run-time option

The run-time options allow you to control certain aspects of your program's processing. For information about run-time options, see the *LE Programming Guide*. There are specific environment variables needed to start the server.

Environment Variables

The LE run-time environment variables are:

NLSPATH
This variable controls where the message catalog is located.

LANG
This variable determines the language of the message catalog. Possible values are **C** (English) and **Ja_JP** (Japanese). The %L parameter on the NLSPATH variable resolves to the value on the LANG variable.

PATH This variable is the search path available to CGI programs. Set the default path for CGI programs.

TZ This variable must be set to use local time in log files and server-side includes.

SHELL
This variable is the name of the default shell program to be run.

LIBPATH
This variable determines the search path for loading DLLs.

The following LE run-time option is used when starting the server:

ENVAR

The ENVAR option sets initial values for specified environment variables that the server is started with. Using ENVAR, you can pass switches or tagged information using standard z/OS UNIX functions. You may set additional environment variables using the _CEE_ENVFILE ENVAR option. The default _CEE_ENVFILE shipped with the server is /etc/httpd.envvars, which is compiled into the IMWHTTPD program. You can use the _CEE_ENVFILE ENVAR option to override this default.

The format of this option is:

```
ENVAR("_CEE_ENVFILE=filename")
```

filename specifies the file containing the LE run-time environment variables. The following is an example of the file containing the environment variables that is shipped with the server under /usr/lpp/internet/server_root/samples/config:

```
SHELL=/bin/sh
PATH=/bin:.:/usr/lpp/internet/bin
LANG=C
```

```
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lpp/internet/%N
TZ=EST5EDT
LIBPATH=/usr/lpp/internet/bin:/usr/lpp/internet/sbin
```

To make the above example reflect Japanese, specify the following:

```
LANG=Ja_JP
LC_ALL=Ja_JP.IBM-939
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lpp/internet/%N
```

Setting server environment variables

Server environment variables can be set from JCL or the OMVS shell. The following steps provide instruction on how to do this.

1. Set variables:

   From JCL: LE Runtime Parameter ENVAR For example:

   ```
   // EXEC PGM=IMWHTTPD,
   // PARM=(ENVAR("MVSDS_CFG=/etc/mycfg"))
   ```

   From OMVS shell:

   ```
   export MVSDS_CFG=/etc/mycfg
   ```

2. LE checks the environment for the variable _CEE_ENVFILE. If it is defined, it MUST contain the name of a file containing additional environment variables to add to the environment. If it is not defined, IMWHTTPD sets it to /etc/httpd.envvars.
3. LE adds each variable defined in the file identified in step 2 to the server's environment. A sample file is shipped with the Web server.

The example shows how to set the variables to preload MVS data sets using the MVS data set configuration file.

-AE
ApplicationEnvironmentName. Identifies the workload management queue. This is only valid on a server address space.

-B *bounce*
The server normally does not bind to its listen ports with the SO_REUSEADDR socket option. This helps to prevent running multiple instances of the server with the same Pid, log, and proxy cache files. When a server is shut down or terminates abnormally, there may be sockets remaining in TIMED_WAIT state in the TCP/IP stack. The Web server retries the bind to its listen ports for up to two minutes to allow previously used sockets to close.

If you know that the prior instance of the server has terminated, you can use the -B flag to set SO_REUSEADDR on to the servers listen ports before binding to them. This avoids the TIMED_WAIT delay. This flag is expecially useful if you have automation software that recognizes Web server termination and needs to start a replacement Web server as quickly as possible.

-drainsoc
Causes the server to drain TCP/IP sockets of all input data before closing the socket. This means that all input data still outstanding on the socket after the server has processed all the data it expects, will be read and discarded. If web server tracing of TCP/IP communications is also being performed, any extra data read will be traced. This can be especially helpful in diagnosing potential browser software problems.

-fscp *FileCodepage*
The FileCodepage is the name of the file system codepage used in codepage conversion when processing text document bodies. When an unknown

codepage is set, the default is used. The default is IBM-1047. Specifying **-fscp** on the httpd command overrides the default set in the server configuration file (httpd.conf).

-netcp *NetCodepage*
The NetCodepage is the network codepage name used in codepage conversion when processing text document bodies. When an unknown codepage is set, the default is used. The default is ISO8859-1 if the -netcp option is not specified. Specifying **-netcp** on the httpd command overrides the default set in the server configuration file (httpd.conf).

Note: The system iconv() service MUST support conversion between the pair of codepages specified as DefaultFsCp and DefaultNetCp.

Some acceptable values are:

netcp	**fscp**
ISO8859-1	IBM-1047
IBM-932C	IBM-939
IBM-eucJC	IBM-939

-gc_only
Only does garbage collection and then exits. This flag is used only for caching proxy servers.

-h HostName. Host names or IP number addresses can be used on the template.

-lb
Listen backlog. Use this parameter to override the default parameter on listen() for master and SSL sockets. If the parameter is greater than SOMAXCONN, stack uses SOMAXCONN instead. The Internet Connection Secure Server 2.1 default is 16; for the Internet Connection Secure Server 2.2 and later versions of the Web server, the default is 128.

-nobg
Runs the server as a foreground process. Do not run the server as a background process.

-nolastmod
LastModified headers are not added to CGI output if they are not already present. The LastModified header affects whether or not browsers will cache the page.

-nosec
Specifies the flag to use to force a base server. Security loads are bypassed.

-nosmf
Turns SMF recording off.

-nosnmp
Turns SNMP support off.

-p *port-number*
Listens on this port number. The default port number is 80. This flag overrides the Port directive specified in the configuration file.

-r *configuration-file*
Specifies the file to use as the configuration file. You must use this flag if you want to start the server with a configuration file other than the default configuration file.

Note: If you specify the -r option and either the normalmode or sslmode option, you **must** specify normalmode or sslmode before -r.

-restart
Restarts a server that is currently running. The httpd command gets the process number of the server that is running from the PidFile and sends the server process ID a SIGHUP signal. The server process after intercepting the SIGHUP signal reloads its configuration file, reopens its log files, and resumes servicing client request. Two instances of the server MUST not be run at the same time using the same PidFile, log files, and proxy cache to avoid corruption.

Because the http daemon must read the configuration file the server is currently using in order to access the PidFile, you must specify the same configuration file when restarting. If you used the **-r** flag and a specific configuration file when you started the server, then you must specify this flag and same file with **-restart**.

-smf
Turns SMF recording on.

-SN
SubsysName. In a workload management environment, each scalable server environment needs a unique name.

-snmp
Turns SNMP support on.

-sslmode [on|off]
For a secure server, turns on the SSL protocol.

-sslport [port]
For a secure server, sets the port used for the SSL protocol.

-version
Returns the version number of the httpd executable and then exits.

Tracing options

The level of tracing provided is -v trace, first level; -vv trace, second level; -mtv, third level; and -debug for maximum tracing. To trace caching, use the -vc option.

Notes:

1. We recommend that you turn tracing on only if instructed to do so by IBM support personnel. IBM support personnel can give you guidance on the most appropriate trace for your problem.
2. To turn tracing off, use the -nodebug option.
3. The –v, –vv, and –mtv flags represent a progression of increasing trace verbosity. They do not need to be used in combination with each other. If more than one of these flags is set, the first flag encountered sets the initial verbosity. All other flags are ignored.
4. The –debug and –nodebug flags are read from left to right by the Web server. If either flag is set without any modules specified, it should be set before the flag with modules is specified. Otherwise, the effect of the flag without the modules will be ignored.

   Example: Set flags as follows:

   ```
   httpd -debug -nodebug stacks
   ```
5. The order of placement of the –debug and –nodebug flags in the command line is independent of the placement of the –v, –vv, and –mtv flags. The –v, –vv, and

–mtv flags are processed before the –debug and –nodebug flags. So, the –debug and –nodebug flags will always override the tracing state set by –v, –vv, and –mtv.

-v Verbose. Turns on first level debugging messages.

Verbose tracing provides basic tracing information on all requests without significantly impacting Web server performance. The design of the fields supports automatic parsing of the information. The format of the trace follows:

Table 7. Verbose trace format

Characters	Field
1-39	Thread Identifier and Timestamp :
40	Space
41	'V' (indicates that this is a first-level tracing line)
42	Space
43-57	Client IP Address
58	Space
59-63	Port
64	Space
65-68	Request Method
69	Space
70-79	Content Length
80	Space
81-83	Response Code
84	Space
85-281	Event Timestamps
282	Space
283-on	Request URI

The `Event Timestamps` are given in elapsed seconds since January 1, 1970. The Web server lists these timestamps as entry-exit pairs, except for the `Request Completion Timestamp`. The elapsed time spent in each module is determined by calculating the difference between the timestamps. The eleven timestamp fields follow:

```
Acceptwait entry/exit
SSL Handshake entry/exit
Request Parsing entry/exit
Service Exit entry/exit
CGI Exit entry/exit
Request Completion Timestamp
```

Example:

```
  9E81518 01/Jun/2001:10:12:26.293840 : V       9.3.7.141    80
 GET      21239 200 0000000000.000000 0000000000.000000 00000000
00.000000 0000000000.000000 0991404746.080941 0991404746.081107
0000000000.000000 0000000000.000000 0000000000.000000 0000000000
.000000 0991404746.293835 /Admin/lgmast.gif
```

The example appears on a single line in the –v trace, even though it is shown here on multiple lines.

-vv
Very Verbose. Turns on second level debugging messages.

-mtv
Much Too Verbose. Turns on third level debugging messages.

-debug [*module_name*]
Turns on debugging for all modules or a specific module. **-debug** with no module name specified turns on debug for all modules.

-nodebug [*module_name*]
Turns off tracing for all modules or a specific module. **-nodebug** with no module name specified turns off debug for all modules.

-vc
Verbose cache. Turns on cache tracing messages.

Module names: On the -debug and -nodebug options, you can enter the following values for *module_name*:

bag
bags
cgi
config
connections
creators
dirbrw
dns
error
errors
fastcgi
format
frca
ftp
gc
gopher
groups
hash
hashtables
html
http
icapi
if
javelin
javelinbase
javelincfg
javelinpics
ldap
ldapv
lex
log
logging
mempool
mpool
mpoolv
netmonitor
nls
none

pics
proxy
proxycache
proxygroup
proxylock
proxylocks
request
resphdrs
ruledb
rules
selftest
ssi
stacks
status
stringlib
tcp
tcpip
threadinit
threadpool
time
timers
users
wbi
workqueue
workthread

Signal handling

The following sections explain how the server behaves when sent the SIGKILL, SIGTERM, and SIGHUP signals:

SIGKILL
This causes the server to terminate immediately. This can be accomplished by sending the SIGKILL signal to the httpd process using the z/OS UNIX shell command kill, the Web server wwwcmd command, or by cancelling the job using the z/OS operator console CANCEL command.

SIGTERM
This causes the server to stop and exit immediately. This can be accomplished by sending the SIGTERM signal to the httpd process using the z/OS UNIX shell command kill or the Web server wwwcmd command.

SIGHUP
Restart a running httpd. This causes httpd to stop accepting new requests, complete current requests, and, if there are no errors, reload the configuration file and resume processing. This can be accomplished by sending the SIGHUP signal to the httpd process using the z/OS UNIX shell command kill or the Web server wwwcmd command. If there are errors, you must fix the configuration file and try again.

The user ID that sends SIGTERM, SIGKILL, and SIGHUP signals to the httpd process must either be the user ID that started the Web server or a superuser ID. You need to know the Web server process ID in order to use the z/OS UNIX shell command, kill, or the Web server wwwcmd command to send the signal. The process ID is defined on the PidFile directive in the configuration file. For information about the kill command, see the *z/OS UNIX System Services Command Reference*.

Examples

- To start the server on port 8080, using the /usr/etc/httpd.conf configuration file instead of the default, /etc/httpd.conf, the following needs to be added to your PROC:

  ```
  // PARM='/-r /usr/etc/httpd.conf -p 8080'
  ```

  If the Port directive is given in the configuration file, the **-p** flag is not required. The **-p** flag can be used to override the value set in the configuration file.
- To start the server using the default configuration file /etc/httpd.conf with verbose tracing:

  ```
  // PARM='/-vv'
  ```
- To start IMWHTTPD using its default configuration file /etc/httpd.conf, from the MVS console, enter:

  ```
  S IMWEBSRV
  ```

IMWIWM PROC (workload management)

The workload management PROC should be placed in your system PROCLIB. This PROC correlates with any query server automatically started from WLM. The passed parameters are set with WLM pane1s as shown in the "Modify an Application Environment" panel on page 232. Ensure that this PROC is properly recognized by your system security.

```
//IMWIWM  PROC IWMSN=,IWMAE=
//*******************************************************************
//*
//* ICSPARM ==> HTTP Server parameters
//* # WLM ApplEnv Queue Server
//* ICSPARM='-SN WEBSN1 -AE WEBHTML'
//*
//*  HTTP Server Parameters:
//*   -SN                        # WLM - subsystem name
//*   -AE                        # WLM - Application Envirnonment
//*
//*******************************************************************
//  SET SN='-SN '
//  SET AE=' -AE '
//  SET QQ='
//WEBSRV   EXEC PROC=IMWEBSRV,REGION=0K,TIME=NOLIMIT,
//   ICSPARM=&QQ&SN&IWMSN&AE&IWMAE&QQ
```

Figure 4. IMWIWM PROC Example

Parameters

-AE
ApplicationEnvironmentName. Identifies the workload management queue. This is only valid on a server address space.

-SN
SubsysName. In a workload management environment, each scalable server environment needs a unique name.

wwwcmd command

Use the wwwcmd command to stop, kill, or restart the server.

Syntax

```
wwwcmd [parameters] action
```

Parameters

Parameters are optional and must be one of the following:

-p *pid*
Specifies the process ID of the server. (For example: -p 1111111)

-f *pid_file*
Specifies the process ID file of the server.

-r *rule_file*
Specifies the rule file to determine the location of the process ID file of the server. The program looks for two directives **ServerRoot** and **PidFile**. If **PidFile** is found, its value is used. If **PidFile** is not found, but **ServerRoot** is found, it is assumed that a file named httpd-pid exists in **ServerRoot**. If neither directive is found, the program uses /etc/httpd.conf as the rule file to search. The default used by the program as the rule file is /etc/httpd.conf.

Action

One action must be specified to the wwwcmd command from the following list:

-stop
This action stops the server gracefully. The server stops accepting any new connections and allows all existing connections to end gracefully before terminating the server. The signal, SIGTERM, gets sent to the server.

-kill
This action stops the server immediately. The server terminates at once, with no clean up or processing of existing connections. It is not recommended that you use this action. You should attempt -stop before entering this command. The signal, KILL, gets sent to the server.

-restart
This action restarts the server. The server stops processing any new connections and allows all existing connections to end gracefully before re-reading the current rule file. The signal SIGHUP gets sent to the server.

Note: You must have permission to send the requested signal to the server. To have permission, you must either log onto the user ID of the server or be the root. IBM recommends you enter this command as root.

z/OS MODIFY console command

You can use the z/OS MODIFY command to:
- Restart the Web server
- Display configuration information and statistics for the Web server
- Turn System Management Facilities (SMF) on or off
- Turn tracing on or off

For more information about the z/OS MODIFY command, see the *z/OS System Command Reference*.

Syntax

```
F [server_jobname],APPL=parameters
```

Parameters

-? Displays the parameters available when using the MODIFY command.

-?? Displays the parameters available on the MODIFY command.

- **Restart parameter:**

  -restart
  Restarts the server after configuration changes.

- **Configuration information and statistics:**

  -d config
  Displays configuration information.

  -d stats
  Displays server statistics.

  -version
  Displays the version of the Web server.

- **SMF parameters:**

  -smf
  Turns SMF recording ON for performance and configuration record data.

  -smf perf
  Turns SMF recording ON for performance record data only.

  -smf config
  Turns SMF recording ON for configuration record data only.

  -? smf
  Displays information about the -smf parameter on the MODIFY command.

  -nosmf
  Turns SMF recording OFF for performance and configuration record data.

  -nosmf perf
  Turns SMF recording OFF for performance record data only. This parameter does not affect configuration data recording by SMF.

  -nosmf config
  Turns SMF recording OFF for configuration record data only. This parameter does not affect the performance data recording by SMF.

  -? nosmf
  Displays information about the -nosmf parameter on the MODIFY command.

Tracing parameters

The level of tracing provided is -v trace, first level; -vv trace, second level; -mtv, third level; and -debug for maximum tracing. To trace caching, use the -vc option.

Notes:

1. We recommend that you turn tracing on only if instructed to do so by IBM support personnel. IBM support personnel can give you guidance on the most appropriate trace for your problem.
2. To turn tracing off, use the -nodebug option.
3. The –v, –vv, and –mtv flags represent a progression of increasing trace verbosity. They do not need to be used in combination with each other. If more than one of these flags is set, the first flag encountered sets the initial verbosity. All other flags are ignored.

4. The –debug and –nodebug flags are read from left to right by the Web server. If either flag is set without any modules specified, it should be set before the flag with modules is specified. Otherwise, the effect of the flag without the modules will be ignored.

   Example: Set flags as follows:

   ```
   httpd -debug -nodebug stacks
   ```

5. The order of placement of the –debug and –nodebug flags in the command line is independent of the placement of the –v, –vv, and –mtv flags. The –v, –vv, and –mtv flags are processed before the –debug and –nodebug flags. So, the –debug and –nodebug flags will always override the tracing state set by –v, –vv, and –mtv.

-v Verbose. Turns on first level tracing to stderr.

Verbose tracing provides basic tracing information on all requests without significantly impacting Web server performance. The design of the fields supports automatic parsing of the information. The format of the trace follows:

Table 8. Verbose trace format

Characters	Field
1-39	Thread Identifier and Timestamp :
40	Space
41	'V' (indicates that this is a first-level tracing line)
42	Space
43-57	Client IP Address
58	Space
59-63	Port
64	Space
65-68	Request Method
69	Space
70-79	Content Length
80	Space
81-83	Response Code
84	Space
85-281	Event Timestamps
282	Space
283-on	Request URI

The `Event Timestamps` are given in elapsed seconds since January 1, 1970. The Web server lists these timestamps as entry-exit pairs, except for the `Request Completion Timestamp`. The elapsed time spent in each module is determined by calculating the difference between the timestamps. The eleven timestamp fields follow:

```
Acceptwait entry/exit
SSL Handshake entry/exit
Request Parsing entry/exit
Service Exit entry/exit
CGI Exit entry/exit
Request Completion Timestamp
```

Example:

```
  9E81518 01/Jun/2001:10:12:26.293840 : V       9.3.7.141    80
 GET      21239 200 0000000000.000000 0000000000.000000 00000000
00.000000 0000000000.000000 0991404746.080941 0991404746.081107
0000000000.000000 0000000000.000000 0000000000.000000 0000000000
.000000 0991404746.293835 /Admin/lgmast.gif
```

The example appears on a single line in the –v trace, even though it is shown here on multiple lines.

-vv
Very Verbose. Turns on second level tracing to stderr.

-mtv
Much Too Verbose. Turns on third level tracing to stderr.

-debug [*module_name*]
Turns on debugging for all modules or a specific module. **-debug** with no module name specified turns on debug for all modules.

-? debug
Displays information about the -debug parameter on the MODIFY command.

-nodebug [*module_name*]
Turns off tracing for all modules or a specific module. **-nodebug** with no module name specified turns off debug for all modules.

-? nodebug
Displays information about the -nodebug parameter on the MODIFY command.

-vc
Verbose cache. Turns on cache tracing messages.

Module names

On the -debug and -nodebug options, you can enter the following values for *module_name*:

bag
bags
cgi
config
connections
creators
dirbrw
dns
error
errors
fastcgi
format
frca
ftp
gc
gopher
groups
hash
hashtables
html
http
icapi
if

javelin
javelinbase
javelincfg
javelinpics
ldap
ldapv
lex
log
logging
mempool
mpool
mpoolv
netmonitor
nls
none
pics
proxy
proxycache
proxygroup
proxylock
proxylocks
request
resphdrs
ruledb
rules
selftest
ssi
stacks
status
stringlib
tcp
tcpip
threadinit
threadpool
time
timers
users
wbi
workqueue
workthread

Examples

To turn SMF recording off, turn debugging for configuration-related functions on, and restart the Web server, enter:

```
F IMWEBSRV,APPL=-nosmf -debug config -restart
```

The following messages are displayed:

```
IMW03514I SMF recording has been disabled for all record types
IMW03504I Debug has been enabled for module, "config"
IMW03537I SA 1207959570 0.0.0.0:8480 * * RESTARTING
IMW03538I SA 1207959570 0.0.0.0:8480 * * RESTART SUCCESSFUL
```

To display information on the Web server version, configuration, and statistics, enter:

```
F IMWEBSRV,APPL=-version -d config -d stats
```

The following example shows messages displayed. For more information on the statistics in IMW03501I, see "Web server activity statistics" on page 249.

```
IMW03516I Version:IBM HTTP Server - North American Edition V 5R2M0
IMW03501I Config: Hostname: host52.raleigh.ibm.com, Port: 80, SSL Port:4480,
 Server root:/usr/lpp/internet/server_root
Tracing all modules.
SMF recording is currently disabled.
IMW3502I Stats: Threads running: 39, Threads idle: 36,
Requests: 19, Bytes rcvd: 15417, Bytes sent: 93109,
Actv In Conns: 3, Actv Out Conns: 0.
Connections since last SMF: 11,
DNS Max: 0.000745, DNS Min: 0.000005, DNS Avg: 0.000038,
Service Plugins Max: 0.035242, Service Plugins Min: 0.019390
Service Plugins Avg: 0.027316,
CGI Max: 15.142639, CGI Min: 12.545370, CGI Avg: 13.844004,
SSL Handshake Max: 199.438877, SSL Handshake Min: 0.000061,
SSL Handshake Avg: 6.657594,
Proxy Response Max: 12.966245, Proxy Response Min: 1.120083,
Proxy Response Avg: 5.128365
Non-SSL Waiting Threads: 14, SSL Waiting Threads: 16,
Async I/O Waiting Threads: 0, Msg Queue Waiting Threads: 0
```

z/OS Workload Management console commands

There are several z/OS commands that are useful when you are working in a Workload Management (WLM) environment. This section describes those commands. For more information on WLM commands, see *z/OS Programming: Workload Management Services*.

WLM mode

If your mode is GOAL, WLM is up and running to manage your system. If your mode is COMPAT, WLM is not activated to manage your system.

Note: You must be in GOAL mode to use the Web server in Scalable Server mode.

To change your mode to GOAL or COMPAT, enter:

```
F WLM,MODE=GOAL
F WLM,MODE=COMPAT
```

Examples:

```
F WLM,MODE=GOAL
IWM007I SYSTEM MVS157 NOW IN WORKLOAD MANAGEMENT GOAL MODE
```

```
F WLM,MODE=COMPAT
SET IPS=00
SET ICS=00
IWM007I SYSTEM MVS157 NOW IN WORKLOAD MANAGEMENT COMPATIBILITY MODE
IEE252I MEMBER IEAIPS00 FOUND IN SYS1.PARMLIB
IEE536I IPS      VALUE 00 NOW IN EFFECT
IEE252I MEMBER IEAICS00 FOUND IN SYS1.PARMLIB
IEE536I ICS      VALUE 00 NOW IN EFFECT
```

Application environments

To check WLM application environments, enter:

```
D WLM,APPLENV=application_environment_name
```

Example:

```
D WLM,APPLENV=WEBHTML
IWM029I  18.42.09  WLM DISPLAY 143
  APPLICATION ENVIRONMENT NAME     STATE     STATE DATA
  WEBHTML                          AVAILABLE
  ATTRIBUTES: PROC=WEBWLM01 SUBSYSTEM TYPE: IWEB
```

To change the status of an Application Environment, enter:

```
VARY WLM,APPLENV=WEBHTML,RESUME
```

Example:

```
VARY WLM,APPLENV=WEBHTML,RESUME
  IWM032I VARY RESUME FOR WEBHTML COMPLETED
```

To activate changes made to ApplEnv panels within WLM when the ApplEnv has not been stopped, enter:

```
VARY WLM,APPLENV=WEBHTML,REFRESH
```

Example:

```
VARY WLM,APPLENV=WEBHTML,REFRESH
  IWM032I VARY REFRESH FOR WEBHTML COMPLETED
```

WLM systems

To check WLM systems, enter:

```
D WLM,SYSTEMS
```

Example:

```
D WLM,SYSTEMS
  IWM025I  18.42.53  WLM DISPLAY 145
  ACTIVE WORKLOAD MANAGEMENT SERVICE POLICY NAME: VICOM1
  ACTIVATED: 1996/11/18  AT: 09:47:41  BY: USER42    FROM: MVS157
  DESCRIPTION: Default VICOM policy with ResGrp
  RELATED SERVICE DEFINITION NAME: COEFFS
  INSTALLED: 1996/11/18  AT: 09:47:20  BY: USER42    FROM: MVS157
  WLM VERSION LEVEL: LEVEL003
  *SYSNAME*  *MODE*  *POLICY*  *WORKLOAD MANAGEMENT STATUS*
  MVS157     GOAL    VICOM1    ACTIVE
```

Appendix B. Configuration directives

Overview of directives 425
Directives that require a stop and restart of the server 425
Directive quick reference 426
Directive syntax guidelines 434
Specifying *request-template* values 434
Specifying a positive or negative string . . . 434
Specifying time 435
Using an asterisk (*) as a wildcard character 435
Using an at sign (@) in user IDs 435
Using a blank or number sign in templates 435
Access control - Set up access control for the server 435
DefProt - Specify default protection setup for requests that match a template 436
Examples: 438
Program default setting 439
Initial configuration file setting 439
Protect - Activate protection setup for requests that match a template 439
Examples: 441
Initial configuration file setting 443
Program default setting 443
Protection - Define a named protection setup within the configuration file 443
Example 444
Initial configuration file setting 444
Program default setting 444
Protection subdirectives 444
ACLOverride - Specify that ACL files override protection setups 445
AuthType - Specify authentication type . . . 445
DeleteMask - Specify the user names, groups, and addresses allowed to delete files . . . 445
GetMask - Specify the user names, groups, and addresses allowed to get files 445
GroupFile - Specify the location of the associated group file 445
Mask - Specify the user names, groups, and addresses allowed to make HTTP requests . 446
PasswdFile - Specify the location of the associated password file 446
PostMask - Specify the user names, groups, and addresses allowed to post files 447
PutMask - Specify the users names, groups, and addresses allowed to put files 447
ServerID - Specify a name to associate with the password file 447
UserID - Specify the Access Control user ID that the server should use 448
SSL client authentication subdirectives . . . 449
Basic - Specify required settings 449
BindSpecific - Specify if the server binds to one or all IP addresses 449
Example 450
Initial configuration file setting 450
Program default setting 450
Bounce — Specify the default start option for the sockets setting SO_REUSEADDR 450
Example 450
Initial configuration file setting 450
Program default setting 450
DNS-Lookup - Specify whether you want to look up host names of clients 450
Example 451
Initial configuration file setting 451
Program default setting 451
HostName - Specify the fully qualified domain name or IP address for the server 451
Example 451
Initial configuration file setting 451
Program default setting 451
imbeds - Specify whether server-side includes will be dynamically imbedded 451
Initial configuration file setting 452
Program default setting 452
InstallPath - Specify an alternate directory installation path 452
Example 453
Initial configuration file setting 453
Program default setting 453
NoLastMod - Specify whether LastModified HTTP headers are added to CGI program output 453
Example 453
Initial configuration file setting 453
Program default setting 453
PidFile - Specify the location of the process ID file 453
Example 454
Initial configuration file setting 454
Program default setting 454
Port - Specify the port on which you want the server to listen for requests 454
Example 454
Initial configuration file setting 454
Program default setting 454
Recovery — Customize ABEND recovery performed by the Web server 454
Initial configuration file setting 455
Program default setting 455
ServerRoot - Specify the current working directory of the server 456
Example 456
Initial configuration file setting 456
ServerToken - Specify whether you want the server to send server identifying information . . 456
Example 456
Initial configuration file setting 456
Program default setting 456
SysDumpName - Specify the high-level qualifier of the IEATDUMP data set 456
Example 457

Initial configuration file setting 457
Program default setting 457
Userid - Specify the Default Access Control user ID 457
Example 458
Initial configuration file setting 458
Codepages - Specify default codepage environment 458
DefaultFsCp - Specify server codepage 458
Example 459
Initial configuration file setting 459
Program default setting 459
DefaultNetCp - Specify codepage. 459
Example 459
Initial configuration file setting 459
Program default setting 459
Directories and Welcome Page - Set viewing options 459
AddBlankIcon - Specify the icon URI used to align the heading of directory listings 460
Example 460
Initial configuration file setting 460
AddDirIcon - Specify the icon URI for directories on directory listings 460
Example 460
Initial configuration file setting 461
AddIcon - Bind an icon to a MIME content-type or encoding-type 461
Example 461
Initial configuration file setting 461
AddParentIcon - Specify the icon URI for a parent directory on directory listings 461
Example 462
Initial configuration file setting 462
AddUnknownIcon - Specify the icon URI for unknown file types on directory listings . . . 462
Example 462
Initial configuration file setting 462
AlwaysWelcome - Specify if a welcome file is returned for all directory requests 462
Example 463
Initial configuration file setting 463
Program default setting 463
DirAccess - Control directory listings 463
Examples: 463
Initial configuration file setting 463
DirReadme - Control directory README files 463
Examples: 463
Initial configuration file setting 464
DirShowBrackets - Use brackets around alternative text on directory listings 464
Example 464
Initial configuration file setting 464
DirShowBytes - Show byte count for small files on directory listings 464
Example 464
Initial configuration file setting 464
DirShowCase - Use case when sorting files on directory listings 464
Example 464
Initial configuration file setting 464
DirShowDate - Show date last modified on directory listings 464
Example 464
Initial configuration file setting 464
DirShowDescription - Show descriptions for files on directory listings 465
Example 465
Initial configuration file setting 465
DirShowGroup - Show the group ID of files on directory listings 465
Example 465
Initial configuration file setting 465
Program default setting 465
DirShowHidden - Show hidden files on directory listings 465
Example 465
Initial configuration file setting 465
DirShowIcons - Show icons in directory listings 465
Example 465
Initial configuration file setting 465
DirShowMaxDescrLength - Set the maximum description length on directory listings 466
Example 466
Initial configuration file setting 466
DirShowMaxLength - Set the maximum length for file names on directory listings 466
Example 466
Initial configuration file setting 466
DirShowMinLength - Set the minimum length for file names on directory listings 466
Example 466
Initial configuration file setting 466
DirShowMode - Show file permissions on directory listings 466
Example 466
Initial configuration file setting 466
Program default setting 466
DirShowOwner - Show file owner on directory listings 466
Example 467
Initial configuration file setting 467
Program default setting 467
DirShowSize - Show file size on directory listings 467
Example 467
Initial configuration file setting 467
IconPath - Specify the path for the directory listing internal icons 467
Example 467
Initial configuration file setting 467
Program default setting 467
Welcome - Specify names of welcome files . . 467
Examples: 468
Initial configuration file setting 469
Program default setting 469
User directories. 469
UserDir - Enable users to have private Web documents 469
Example 469
Default 469

Error messages - Customize Web server error messages . . . 469
ErrorPage - Specify a customized message for a particular error condition . . . 470
Example . . . 470
Initial configuration file setting . . . 470
Program default setting . . . 470
Error conditions, causes, and default messages . . . 471
Log401Error - Specify whether IMW0196I NOT AUTHENTICATED should be written to the error log . . . 476
Example . . . 476
Initial configuration file setting: . . . 476
Program default setting: . . . 477
GWAPI - Specify GWAPI applications for processing . . . 477
Customize Web server process steps . . . 477
ServerInit - Customize the initialization step 477
WLMClassify - Customize the WLM pre-exit step . . . 478
PreExit - Customize the pre-exit step . . . 478
NameTrans - Customize the Name Translation step . . . 478
Authorization - Customize the Authorization step . . . 479
Authentication - Customize the Authentication step . . . 480
ObjectType - Customize the Object Type step 480
Service - Customize the Service step . . . 481
PICSDBLookup - Customize the PICS label retrieval step . . . 482
DataFilter - Customize the Data Filter step 482
Log - Customize the Log step . . . 483
Error - Customize the Error step . . . 483
PostExit - Customize the post-exit step . . . 484
ServerTerm - Customize the server termination step . . . 484
CounterDirectory - Specify a directory for the HTCounter program . . . 484
Examples . . . 485
Initial configuration file setting . . . 485
Program default setting . . . 485
DebugToolAddr - Identify the workstation running the Remote Debugger . . . 485
Example . . . 485
Initial configuration file setting . . . 485
Lightweight Directory Access Protocol - Set up shared configuration for the server . . . 485
LDAPInfo - Define an external LDAP server . . . 485
Example . . . 486
Initial configuration file setting . . . 486
Program default setting . . . 486
LDAPInfo Subdirectives . . . 486
General subdirectives . . . 486
Timeout subdirectives . . . 487
Server connection subdirectives . . . 487
Client connection subdirectives . . . 488
LDAPInclude - Retrieve configuration file information from the LDAP server . . . 490
Format . . . 490
Example . . . 491
Initial configuration file setting . . . 491
Program default setting . . . 491
Logging and Reporting - Customize logs and generate reports . . . 491
DoReporting — Use logging and reporting options . . . 491
Example . . . 492
Initial configuration file setting . . . 492
Program default setting . . . 492
AccessLog - Name the path for the access log file . . . 492
Example . . . 492
Initial configuration file setting . . . 492
Program default setting . . . 492
AccessLogArchive - Remove existing access, agent, or referer log files or run a user exit . . . 493
Examples: . . . 493
Initial configuration file setting . . . 494
Program default setting . . . 494
AccessLogExcludeURL - Suppress log entries for specific files or directories . . . 494
Example . . . 494
Initial configuration file setting . . . 494
Program default setting . . . 494
AccessLogExcludeMethod - Suppress log entries for files or directories requested by a given method . . . 494
Example . . . 494
Initial configuration file setting . . . 494
Program default setting . . . 494
AccessLogExcludeMimeType - Suppress log entries for specific MIME types . . . 495
Example . . . 495
Initial configuration file setting . . . 495
Program default setting . . . 495
AccessLogExcludeReturnCode - Suppress log entries for specific return codes . . . 495
Example . . . 495
Initial configuration file setting . . . 495
Program default setting . . . 495
AccessLogExpire - Remove existing access log files when they reach a given age in days . . . 495
Example . . . 496
Initial configuration file setting . . . 496
Program default setting . . . 496
AccessLogSizeLimit - Remove existing access log files when they reach a given collective size . 496
Example . . . 496
Initial configuration file setting . . . 496
Program default setting . . . 496
AccessReportDescription - Give a short description of the HTLOGREP report to be created . . . 496
Example . . . 497
Initial configuration file setting . . . 497
Program default setting . . . 497
AccessReportDoDnsLookup- Display client hostnames in HTLOGREP access reports . . . 497
Example . . . 497
Initial configuration file setting . . . 497

Program default setting 497
AccessReportExcludeURL - Suppress log entries for specific files or directories from the HTLOGREP report 497
Examples. 497
Initial configuration file setting 498
Program default setting 498
AccessReportIncludeURL - Include only log entries for specific files or directories in the HTLOGREP report 498
Example 498
Initial configuration file setting 498
Program default setting 498
AccessReportExcludeHostName - Suppress the log entries for specific host names from the HTLOGREP report 498
Example 499
Initial configuration file setting 499
Program default setting 499
AccessReportIncludeHostName - Include only log entries for specific host names in the HTLOGREP report 499
Example 499
Initial configuration file setting 499
Program default setting 499
AccessReportExcludeMethod - Suppress the log entries of a given method type from the HTLOGREP report 499
Example 500
Initial configuration file setting 500
Program default setting 500
AccessReportExcludeReturnCode - Suppress the log entries with a given return code from the HTLOGREP report 500
Example 500
Initial configuration file setting 500
Program default setting 500
AccessReportRoot - Name the path for the root directory where HTLOGREP access log reports are stored 500
Example 500
Initial configuration file setting 501
Program default setting 501
AccessReportTemplate - Name the HTLOGREP report template. 501
Example 501
Initial configuration file setting 501
Program default setting 501
AccessReportTopList - Specify the top number of items on which HTLOGREP is to report . . 501
Example 501
Initial configuration file setting 501
Program default setting 502
AgentLog - Name the path for the agent log file 502
Example 502
Initial configuration file setting 502
Program default setting 502
CacheAccessLog - Specify the path for the cache access log files 502
Example 503
Initial configuration file setting 503
Program default setting 503
CgiErrorLog - Name the path for the CGI error log file 503
Example 503
Initial configuration file setting 503
Program default setting 503
ErrorLog - Name the file where you want to log internal server errors 503
Example 503
Initial configuration file setting 504
Program default setting 504
ErrorLogArchive - Remove existing error or CGI error log files or run a user exit 504
Examples: 505
Initial configuration file setting 505
Program default setting 505
ErrorLogExpire - Remove existing error log files when they reach a given age in days 505
Example 505
Initial configuration file setting 505
Program default setting 505
ErrorLogSizeLimit - Remove existing error log files when they reach a given collective size . . 505
Example 506
Initial configuration file setting 506
Program default setting 506
LogFormat - Specify common or old log file format. 506
Example 506
Initial configuration file setting 506
Program default setting 506
LoggingReportingDebugOutput— Generate debug log for HTLOGREP reporting program . 506
Example 506
Initial configuration file setting 507
Example of HTLOGREP Debug Information Log. 507
LoggingReportingProgram — Specify the reporting program to be used 509
Example if using third-party program . . . 509
Example if using HTLOGREP and changing default options 509
Initial configuration file setting 509
Program default setting 509
LoggingReportingProgramOptions — Specify reporting program options 509
Example if using HTLOGREP and changing default options 510
Initial configuration file setting 510
Program default setting 510
LogTime - Specify GMT or local time stamps in log files 510
Example 510
Initial configuration file setting 510
Program default setting 510
LogToSyslog - Log access information to the syslog, in addition to or instead of the error log . 510
Example 511
Initial configuration file setting 511
Program default setting 511

NoLog - Suppress log entries for specific hosts or domains matching a template 511
Example 511
Initial configuration file setting 511
Program default setting 511
ProxyAccessLog - Name the path for the proxy access log file 511
Example 511
Initial configuration file setting 512
Program default setting 512
RefererLog - Name the path for the referer log file 512
Example 512
Initial configuration file setting 512
Program default setting 512
ReportDataCompressionProgram - Specify path to the compression program 512
Example 512
Initial configuration file setting 512
Program default setting 513
ReportDataUnCompressionProgram - Specify path to the uncompression program 513
Example 513
Initial configuration file setting 513
Program default setting 513
ReportDataCompressionSuffix - Specify the suffix appended to the compressed report data files 513
Example 513
Initial configuration file setting 513
Program default setting 513
ReportProcessOldLogs - Check for old logs in the log directory 513
Examples. 514
Initial configuration file setting 514
Program default setting 514
ReportDataSizeLimit - Remove existing access data files when they reach a given collective size. 514
Example 514
Initial configuration file setting 514
Program default setting 514
ReportDataArchive - Specify whether to remove existing accessdata files 514
Examples: 515
Initial configuration file setting 515
Program default setting 515
ReportDataExpire - Remove existing access data files when they reach a given age in days . . . 515
Example 516
Initial configuration file setting 516
Program default setting 516
SMF - Specify the type of information that SMF records 516
Example 516
Initial configuration file setting 516
Program default setting 516
SMFRecordingInterval - Specify how often to record performance record information 516
Example 517
Initial configuration file setting 517
Program default setting 517
Meta-Information - Name meta-information files and directories 517
MetaDir - Specify name of subdirectory for meta-information files 517
Example 518
Initial configuration file setting 518
Initial configuration file setting 518
MetaSuffix - Specify the suffix for meta-information files 518
Example 518
Initial configuration file setting 518
Program default setting 518
Methods - Set method acceptance 518
Disable - Disable HTTP methods 519
Example 519
Initial configuration file setting 519
Enable - Enable HTTP methods 519
Example 520
Example 520
Initial configuration file setting 520
Multi-format processing - Define file extensions for multi-format processing 520
Multi-Format Processing. 520
Summary of resource mapping directives and meta-information 521
Computing file quality 521
AddLanguage - Specify the language of files with particular suffixes 522
Examples. 522
Initial configuration file setting 522
Program default setting 522
AddEncoding - Specify the MIME content encoding of files with particular suffixes . . . 523
Example 523
Initial configuration file setting 523
AddCharSet - Specify the character set documents are encoded in 523
Example 523
Initial configuration file setting 523
AddType - Specify the data type of files with particular suffixes 523
Example 525
Initial configuration file settings 525
AddClient - Specify file extensions for requesting clients 527
Examples. 528
Initial configuration file setting 528
Program default setting 528
Using automatic browser detection for Welcome pages. 529
SuffixCaseSense - Specify whether suffix definitions are case sensitive 529
Example 529
Initial configuration file setting 529
Program default setting 529
Proxy server settings — Configure the Web server as a proxy server 529
CacheDefaultExpiry - Specify a default expiration date 529
Initial configuration file setting 530

Program default setting 530
CacheExpiryCheck - Specify whether the server will return expired files 530
Initial configuration file setting 530
Program default setting 530
CacheLastModifiedFactor - Specify fraction of Last-Modified date to use in determining expiration date 530
Initial configuration file setting 530
Program default setting 530
CacheLimit_2 - Specify upper limit for cached file size 530
Initial configuration file setting 531
Program default setting 531
CacheNoConnect - Specify stand alone cache mode 531
Initial configuration file setting 531
Program default setting 531
CacheOnly - Cache only files with URIs that match a template 531
Example 531
Initial configuration file setting 531
Program default setting 531
CacheRoot - Specify cache root directory . . . 531
Example 532
Initial configuration file setting 532
Program default setting 532
CacheSize - Specify cache size 532
Initial configuration file setting 532
Program default setting 532
CacheUnused - Specify how long to keep unused cached files that match a template. . . 532
Examples. 532
Initial configuration file setting 532
Program default setting 532
Caching - Turn proxy caching off or on. . . . 533
Initial configuration file setting 533
Program default setting 533
ftp_proxy - Specify a proxy server for this proxy to connect to for FTP requests 533
Example 533
Initial configuration file setting 533
Program default setting 533
Gc - Turn garbage collection on or off 533
Initial configuration file setting 533
Program default setting 533
GcDailyGc - Specify a daily time for garbage collection 534
Example 534
Initial configuration file setting 534
Program default setting 534
GcMemUsage - Specify how much memory to use for garbage collection 534
Initial configuration file setting 534
Program default setting 534
gopher_proxy - Specify a proxy server for this proxy to connect to for Gopher requests . . . 534
Example 534
Initial configuration file setting 534
Program default setting 534
http_proxy - Specify a proxy server for this proxy to connect to for HTTP requests 535
Example 535
Initial configuration file setting 535
Program default setting 535
no_proxy - Connect directly to domains matching templates 535
Example 535
Initial configuration file setting 535
Program default setting 535
NoCaching - Do not cache files with URIs that match a template 535
Example 536
Initial configuration file setting 536
Program default setting 536
ProxyAccessLog — Name the path for the proxy access log file 536
SocksServer - Specify a socks server 536
Format 536
Examples. 536
Initial configuration file setting 536
Program default setting 536
Resource mapping - Redirect URLs 536
Exec - Run a CGI program for matching requests 537
Examples. 538
Initial configuration file setting 538
ExecDirPass - Control access to directories matching Exec directives 539
Examples. 539
Program default setting 539
Fail - Reject matching requests 539
Examples. 540
Initial configuration file setting 540
Map - Change matching requests to a new result string 540
Examples. 541
Initial configuration file setting 541
Pass - Accept matching requests 541
Examples. 542
Initial configuration file setting 543
Redirect - Send matching requests to another server 543
Example 544
Initial configuration file setting 544
InheritEnv - Specify which environment variables are inherited by CGI programs . . . 544
Example 545
Initial configuration file setting 545
Program default setting 545
DisInheritEnv - Specify which environment variables are disinherited by CGI programs . . 545
Example 545
Initial configuration file setting 545
Program default setting 545
Security - Set up secure connections for the server 545
KeyFile - Specify the name of a key database or SAF key ring to be used for SSL connections . . 546
Example 546
Initial configuration file setting 546
Program default setting 546

NormalMode - Turn port on or off for HTTP connections 546
Initial configuration file setting 547
Program default setting 547
SSLCipherSpec - Specify the levels of encryption to use for SSL connections 547
Valid codes for *cipher_specification* encryption: 547
Example 548
Initial configuration file setting 548
Program default setting 549
SSLClientAuth - Select the type of SSL client authentication 549
Examples. 549
Initial configuration file setting 550
Program default setting 550
SSLMode - Turn SSL on or off. 550
Initial configuration file setting 550
Program default setting 550
SSLPort - Set port for SSL security 550
Initial configuration file setting 550
Program default setting 550
SSLServerCert - Associate a key database with an IP address 551
Examples. 551
Initial configuration file setting 551
Program default setting 551
Hints and Tips 551
SSLV2Timeout — Set SSL V2 session timeout value 552
Initial configuration file setting 552
Program default setting 552
SSLV3Timeout — Set SSL V3 session timeout value 552
Initial configuration file setting 552
Program default setting 552
SSLX500CARoots — Specify location of trusted CA certificates 553
Initial configuration file setting 553
Program default setting 553
SSLX500Host — Specify host name or IP address of the X.500 directory server 553
Initial configuration file setting 554
Program default setting 554
SSLX500Port — Specify X.500 directory server port number. 554
Initial configuration file setting 554
Program default setting 554
SSLX500UserID — Specify user ID for the LDAP connection to the X.500 directory server . 554
Initial configuration file setting 554
Program default setting 554
SSLX500Password — Specify user ID password for the LDAP connection to the X.500 directory server 555
Initial configuration file setting 555
Program default setting 555
System Management - Define system management settings 555
ApplEnv - Specify application environment for workload management 555
Example 556
Initial configuration file setting 556
ApplEnvConfig - Specify a set of directives to tailor the application environment 556
Example 558
Initial configuration file setting 558
Program default setting 558
ApplEnvMax - Specify maximum number of Queue Servers that WLM is to maintain for a specific application environment 558
Example 558
Initial configuration file setting 558
Program default setting 558
ApplEnvMin - Specify minimum number of Queue Servers that WLM is to maintain for a specific application environment 558
Example 559
Initial configuration file setting 559
Program default setting 559
ApplEnvPrestart - Specify an application environment to start at initialization. 559
Example 559
Initial configuration file setting 560
Program default setting 560
PluginDefault - Specify default Plugin action 560
Example 560
Initial configuration file setting 560
Program default setting 560
PluginExclude - Specify Plugin not to load during initialization 560
Example 560
Initial configuration file setting 560
Program default setting 560
PluginInclude - Specify Plugin to be loaded during initialization 561
Example 561
Initial configuration file setting 561
Program default setting 561
Using configuration directives to tailor your application environments 561
SNMP - Enable and disable SNMP support . . 563
Example 564
Initial configuration file setting 564
Program default setting 564
SNMPCommunityName - Provide a security password for SNMP 564
Example 564
Initial configuration file setting 564
Program default setting 564
WebMasterEmail - Create an e-mail address to receive SNMP problem reports 564
Example 564
Initial configuration file setting 564
Program default setting 564
Timeouts - Close connections automatically . . . 564
InputTimeout - Specify time allowed for the client to send a request 564
Examples. 565
Initial configuration file setting 565
Program default setting 565
OutputTimeout - Specify maximum time for sending output to the client 565

Examples 565
Initial configuration file setting 565
Program default setting 565
ScriptTimeout - Specify time allowed for a program to complete 565
Examples 566
Initial configuration file setting 566
Program default setting 566
Tuning - Define performance and scalability settings 566
CacheLocalFile - Specify files you want to load in memory at start up 566
Example 566
Initial configuration file setting 567
Program default setting 567
CacheLocalMaxBytes - Specify maximum amount of memory to use for file caching . . . 567
Example 567
Initial configuration file setting 567
Program default setting 567
CacheLocalMaxFiles - Specify the maximum number of files for caching 567
Example 567
Initial configuration file setting 567
Program default setting 567
EnableFRCA — Turn dynamic caching on or off 567
Example 568
Initial configuration file setting 568
Program default setting 568
Enclave — Create and join a Workload Management enclave for the current request . . 568
Example 569
Initial configuration file setting 569
Program default setting 569
FRCAAccessLog — Specify a log file path and name for dynamic caching requests 569
Example 569
Initial configuration file setting 569
Program default setting 569
FRCACacheEntries — Specify the maximum number of files to be dynamically cached . . . 570
Example 570
Initial configuration file setting 570
Program default setting 570
FRCACacheOnly — Specify URIs to be dynamically cached 570
Example 570
Initial configuration file setting 570
Program default setting 570
FRCACacheSize — Specify size of the dynamic cache 570
Example 570
Initial configuration file setting 570
Program default setting 570
FRCAMaxFileSize — Specify maximum file size for the dynamic cache 571
Example 571
Initial configuration file setting 571
Program default setting 571
FRCANoCaching — Exclude URIs from the dynamic cache 571
Example 571
Initial configuration file setting 571
Program default setting 571
FRCAStackName — Specify the TCP/IP stack that supports the dynamic cache 571
Example 572
Initial configuration file setting 572
Program default setting 572
FRCAVirtualHost — Indicate to the dynamic cache whether multiple virtual hosts or IP addresses are being used 572
Example 572
Initial configuration file setting 572
Program default setting 572
FRCAWLMParms — Specify parameters for Workload Management 572
Example 573
Initial configuration file setting 573
Program default setting 573
ListenBacklog - Specify the number of listen backlog client connections for the server to carry 573
Example 573
Initial configuration file setting 573
LiveLocalCache - Specify whether the cache is updated when a cached file is modified . . . 573
Initial configuration file setting 573
Program default setting 574
MaxActiveThreads - Specify the maximum number of active threads 574
Example 575
Program default setting 575
Initial configuration file setting 575
MaxContentLengthBuffer - Set the size of the buffer when computing content length 575
Example 575
Initial configuration file setting 575
Program default setting 576
MaxPersistRequest - Specify the maximum number of requests to receive on a persistent connection 576
Initial configuration file setting 576
Program default setting 576
PersistTimeout - Specify time to wait for the client to send another request 576
Examples 576
Initial configuration file setting: 576
Program default setting: 576
QOS - Specify Quality of Service classification information you want to send to TCP/IP on each request 577
Example 577
Initial configuration file setting: 577
Program default setting: 577
ServerPriority - Specify the priority you want your server to have on your system 577
Example 577
Initial configuration file setting 577
UseACLs - Specify whether ACL files will be checked 577
Example 578
Initial configuration file setting 578

Program default setting 578
UseMetaFiles - Specify whether meta files will be used 578
Example 578
Initial configuration file setting 578
Program default setting 578
WLMSN- Enable the creation of request level Workload Management enclaves 578
Example 579
Initial configuration file setting 579
Program default setting 579

Overview of directives

This appendix includes descriptions of directives in the Web server configuration file. The default configuration file name is httpd.conf.

Each directive description includes:

- Heading with the directive name and a brief description
- Usage instructions
- Example of how the directive might appear in the configuration file

  Each configuration directive follows the same general syntax:

  ```
  DirectiveName value
  ```

- Initial configuration file setting

  This is the setting of the directive in the configuration file before it is changed.
- Default value or values of the directive

Directives that require a stop and restart of the server

For most configuration directives, your updates take effect when you restart the Web server. For the following directives, you must stop and restart the server for changes to take effect:

- BindSpecific
- Bounce
- CacheRoot
- CacheSize
- Caching
- DefaultFsCp
- DefaultNetCp
- Enclave
- HostName
- KeyFile
- NormalMode
- PidFile
- Port
- ServerRoot
- SSLCipherSpec
- SSLClientAuth
- SSLMode
- SSLPort
- SSLServerCert
- SSLV2Timeout
- SSLV3Timeout
- SSLX500CARoots
- SSLX500Host

- SSLX500Port
- SSLX500UserID
- SSLX500Password

Directive quick reference

The directive descriptions are grouped according to function. Within each group, the directives are in alphabetical order. For an alphabetical list of directives, see the Directives entry in the index.

Type	Description	Page	Name
Access control	Specify default protection setup for requests that match a template	436	DefProt
	Activate protection setup for requests that match a template	439	Protect
	Define a named protection setup within the configuration file	443	Protection
		444	Protection subdirectives
		449	SSL client authentication subdirectives
Basic	Specify if the server binds to one or all IP addresses	449	BindSpecific
	Specify the default start option for the sockets setting SO_REUSEADDR	450	Bounce
	Specify whether you want to look up host names of clients	450	DNS-Lookup
	Specify the fully qualified domain name or IP address for the server	451	HostName
	Specify whether server-side includes will be dynamically imbedded	451	imbeds
	Specify an alternate directory installation path	452	InstallPath
	Specify whether LastModified HTTP headers are dded to CGI program output	"NoLastMod - Specify whether LastModified HTTP headers are added to CGI program output" on page 453	NoLastMod
	Specify the location of the process ID file	453	PidFile
	Specify the port on which you want the server to listen for requests	454	Port
	Customize ABEND recovery performed by the Web server	454	Recovery
	Specify the absolute directory path of the server	456	ServerRoot

Type	Description	Page	Name
	Specify whether you want the server to send server identifying information in the default error pages.	456	ServerToken
	Specify the high-level qualifier of the IEATDUMP data set.	456	SysDumpName
	Specify the default surrogate user ID	457	UserId
Codepages	Specify the codepages the server receives from the client	459	DefaultNetCp
	Specify the codepage environment variable for the server	458	DefaultFsCp
Directories and Welcome Page	Specify the icon URI used to align the heading of directory listings	460I	AddBlankIcon
	Specify the icon URI for directories on directory listings	460	AddDirIcon
	Bind an icon to a MIME content-type or encoding-type	461	AddIcon
	Specify the icon URI for a parent directory on directory listings	461	AddParentIcon
	Specify the icon URI for unknown file types on directory listings	462	AddUnknownIcon
	Specify if a welcome file is returned for all directory requests	462	AlwaysWelcome
	Control directory listings	463	DirAccess
	Control directory README files	463	DirReadme
	Use brackets around alternative text on directory listings	464	DirShowBrackets
	Show byte count for small files on directory listings	464	DirShowBytes
	Use case when sorting files on directory listings	464	DirShowCase
	Show date last modified on directory listings	464	DirShowDate
	Show descriptions for files on directory listings	465	DirShowDescription
	Show the group ID of files on directory listings	465	DirShowGroup
	Show hidden files on directory listings	465	DirShowHidden
	Show icons in directory listings	465	DirShowIcons
	Set the maximum description length on directory listings	466	DirShowMaxDescrLength
	Set the maximum length for file names on directory listings	466	DirShowMaxLength
	Set the minimum length for file names on directory listings	466	DirShowMinLength
	Show file permissions on directory listings	466	DirShowMode

Type	Description	Page	Name
	Show file owner on directory listings	466	DirShowOwner
	Show file size on directory listings	467	DirShowSize
	Specify the path for the directory listing internal icons	467	IconPath
	Specify names of welcome files	467	Welcome
	Enable users to have private Web documents	469	UserDir
Error message customization	Specify a customized message for a particular error condition	470	ErrorPage
	Specify whether IMW0196I NOT AUTHENTICATED should be written to the error log	476	Log401Error
GWAPI application processing	Customize the server initialization step	477	ServerInit
	Customize the WLM pre-exit step	478	WLMClassify
	Customize the user pre-exit step	478	PreExit
	Customize the authentication step	480	Authentication
	Customize the name translation step	478	NameTrans
	Customize the authorization step	479	Authorization
	Customize the object type step	480	ObjectType
	Customize the service step	481	Service
	Customize the PICS label retrieval step	482	PICSDBLookup
	Customize the data filter step	482	DataFilter
	Customize the log step	483	Log
	Customize the error step	483	Error
	Customize the post-exit step	484	PostExit
	Customize the server termination step	484	ServerTerm
	Identify the workstation running the Remote Debugger	485	DebugToolAddrt
	Specify the directory for the HTCounter program	484	CounterDirectory
LDAP servers	Specify external LDAP server information	485	LDAPInfo
		486	LDAPInfo subdirectives
	Retrieve configuration file information from an LDAP server	490	LDAPInclude
Logging and Reporting	Use logging and reporting options	491	DoReporting
	Name the path for the access log file	492	AccessLog
	Remove existing access log files or run a user exit	493	AccessLogArchive
	Suppress log entries for specific files or directories	494	AccessLogExcludeURL

Type	Description	Page	Name
	Suppress log entries for files or directories requested by a given method	494	AccessLogExcludeMethod
	Suppress log entries for specific MIME types	495	AccessLogExcludeMimeType
	Suppress log entries for specific return codes	495	AccessLogExcludeReturnCode
	Remove existing access log files when they reach a given age in days	495	AccessLogExpire
	Remove existing access log files when they reach a given collective size	496	AccessLogSizeLimit
	Give a short description of the report to be created	496	AccessReportDescription
	Suppress from the report the log entries for specific files or directories	497	AccessReportExcludeURL
	Include in the report only log entries for specific files directories	498	AccessReportIncludeURL
	Suppress from the report the log entries for specific host names	498	AccessReportExcludeHostName
	Include in the report only log entries for specific host names	499	AccessReportIncludeHostName
	Suppress from the report the log entries of a given method type	499	AccessReportExcludeMethod
	Suppress from the report the log entries with a given range of return codes	500	AccessReportExcludeReturnCode
	Name the path for the root directory where access log reports are stored	500	AccessReportRoot
	Name the report template	501	AccessReportTemplate
	Specify the top number of items on which to report	501	AccessReportTopList
	Specify the path for the agent log files	502	AgentLog
	Specify the path for the cache access log files	502	CacheAccessLog
	Specify the path for the CGI error log files	503	CgiErrorLog
	Name the file where you want to log internal server errors	503	ErrorLog
	Remove existing error log files or run a user exit	504	ErrorLogArchive
	Remove existing error log files when they reach a given age in days	505	ErrorLogExpire
	Remove existing error log files when they reach a given collective size	505	ErrorLogSizeLimit
	Specify common or old log file format	506	LogFormat

Type	Description	Page	Name
	Generate debug log for HTLOGREP reporting program	506	LoggingReportingDebugOutput
	Specify the reporting program to be used	491	LoggingReportingProgram
	Specify reporting program options	509	LoggingReportingProgramOptions
	Specify GMT or local time stamps in log files	510	LogTime
	Suppress log entries for specific hosts or domains matching a template	511	NoLog
	Specify the path for the referer access log files	512	RefererLog
	Specify path to the compression program used for the access data files	512	ReportDataCompressionProgram
	Specify path to the uncompression program used for the access data files	513	ReportDataUnCompressionProgram
	Specify the suffix appended to the compressed file	513	ReportDataCompressionSuffix
	Specify how to handle old log files not originally included in a report.	513	ReportProcessOldLogs
	Remove existing access data files when they reach a given collective size. These access data files are named access.*mmddyyyy* and are used for reports.	514	ReportDataSizeLimit
	Specify whether to remove existing access data files. These access data files are named access.*mmddyyyy* and are used for reports.	514	ReportDataArchive
	Remove existing access data files when they reach a given age. These access data files are named access.*mmddyyyy* and are used for reports.	515	ReportDataExpire
	Write configuration and performance data to an SMF memory segment	516	SMF
	Define how often SMF writes performance record information to a file	516	SMFRecordingInterval
Meta-Information	Specify name of subdirectory for meta-information files	517	MetaDir
	Specify the suffix for meta-information files	518	MetaSuffix
Methods	Disable HTTP methods	519	Disable
	Enable HTTP methods	519	Enable
Multi-format processing	Specify the language of files with particular suffixes	522	AddLanguage

Type	Description	Page	Name
	Specify the MIME content encoding of files with particular suffixes	523	AddEncoding
	Specify, for a given suffixes, the codepage in which the character sets documents are stored	523	AddCharSet
	Specify the data type of files with particular suffixes	523	AddType
	Specify whether suffix definitions are case sensitive	529	SuffixCaseSense
	Specify file suffixes to use when responding to particular browsers	527	AddClient
Proxy server settings	Specify a default expiration date	529	CacheDefaultExpiry
	Specify whether the server will return expired files	530	CacheExpiryCheck
	Specify fraction of Last-Modified date to use in determining expiration date	530	CacheLastModifiedFactor
	Specify upper limit for cached file size	530	CacheLimit_2
	Specify stand alone cache mode	531	CacheNoConnect
	Cache only files with URLs that match a template	531	CacheOnly
	Specify cache root directory	531	CacheRoot
	Specify cache size	532	CacheSize
	Specify how long to keep unused cached files that match a template	532	CacheUnused
	Turn proxy caching on/off	533	Caching
	Specify a proxy server for this proxy to connect to for FTP requests	533	ftp_proxy
	Turn garbage collection on or off	533	Gc
	Specify a daily time for garbage collection	534	GcDailyGc
	Specify how much memory to use for garbage collection	534	GcMemUsage
	Specify a proxy server for this proxy to connect to for Gopher requests	534	gopher_proxy
	Specify a proxy server for this proxy to connect to for HTTP requests	535	http_proxy
	Set the size of the buffer when computing content length	575	MaxContentLengthBuffer
	Connect directly to domains matching templates	535	no_proxy
	Do not cache files with URLs that match a template	535	NoCaching
	Name the path for the proxy access log file	511	ProxyAccessLog
	Specify a socks server	536	SocksServer

Type	Description	Page	Name
Resource Mapping	Run a CGI program for matching requests	537	Exec
	Controls access to directories matching Exec directives	539	ExecDirPass
	Reject matching requests	539	Fail
	Change matching requests to a new result string	540	Map
	Accept matching requests	541	Pass
	Send matching requests to another server	543	Redirect
	Specify which environment variables are inherited by CGI programs	544	InheritEnv
	Specify which environment variables are disinherited by CGI programs	545	DisInheritEnv
Security	Register a key database, key file, or key ring with the HTTP Server	546	KeyFile
	Turn port on or off for HTTP connections	546	NormalMode
	Specify levels of encryption supported by the server	547	SSLCipherSpec
	Select the type of SSL client authentication	549	SSLClientAuth
	Turn port on or off for SSL	550	SSLMode
	Set port for SSL security	550	SSLPort
	Associate a key database with an IP address	551	SSLServerCert
	Set SSL V2 session timeout value	552	SSLV2Timeout
	Set SSL V3 session timeout value	552	SSLV3Timeout
	Specify location of trusted CA certificates	553	SSLX500CARoots
	Specify host name or IP address of the X.500 directory server	553	SSLX500Host
	Specify X.500 directory server port number	554	SSLX500Port
	Specify user ID for the LDAP connection to the X.500 directory server	554	SSLX500UserID
	Specify user ID password for the LDAP connection to the X.500 directory server	555	SSLX500Password
System Management	Specify the application environments and workload management transaction classes for processing URL requests matching a given template	555	ApplEnv
	Specify minimum number of Queue Servers that WLM is to maintain for a specific application environment	558	ApplEnvMin

Type	Description	Page	Name
	Specify maximum number of Queue Servers that WLM is to maintain for a specific application environment	558	ApplEnvMax
	Specify an application environment to start at initialization	559	ApplEnvPrestart
	Specify a set of directives to tailor an application environment	556	ApplEnvConfig
	Set default plugin action	560	PluginDefault
	Specify plugin not to be loaded during configuration	560	PluginExclude
	Specify plugin to be loaded during configuration	561	PluginInclude
	Enable and disable SNMP support	563	SNMP
	Provide a security password to control access to a particular community of monitored servers	564	SNMPCommunityName
	Create an e-mail address to receive SNMP problem reports	564	WebMasterEmail
Timeouts	Specify input timeout setting	564	InputTimeout
	Specify output timeout setting	565	OutputTimeout
	Specify script timeout setting	565	ScriptTimeout
Tuning settings	Specify files you want to load in memory at start up	566	CacheLocalFile
	Specify the maximum amount of memory allocated for file caching	567	CacheLocalMaxBytes
	Specify the maximum number of files to be simultaneously cached	567	CacheLocalMaxFiles
	Enable and customize dynamic caching using the Fast Response Cache Accelerator	567	EnableFRCA
	Create and join a Workload Mangement (WLM) enclave for the current request.	568	Enclave
	Specify a log file path and a log file name so that requests served by the Fast Response Cache Accelerator log in a separate log file.	569	FRCAAccessLog
	Specify the maximum number of files you want the server to dynamically cache using the Fast Response Cache Accelerator.	570	FRCACacheEntries
	Specify the number of listen backlog client connections you want the server to carry before sending connection refused messages to clients.	573	ListenBacklog
	Specify whether the cache is updated when a cached file is modified	573	LiveLocalCache

Type	Description	Page	Name
	Specify the maximum number of threads to have active	574	MaxActiveThreads
	Set the size of the buffer when computing content length	575	MaxContentLengthBuffer
	Specify the maximum number of requests the server receives on a persistent connection	576	MaxPersistRequest
	Specify the amount of time the server waits before cancelling a persistent connection	576	PersistTimeout
	Specify what Quality of Service (QOS) classification information is sent to TCP/IP on each request	577	QOS
	Specify the priority you want your server to have on your system	577	ServerPriority
	Specify whether meta files will be checked	577	UseACLs
	Specify whether meta files will be used	578	UseMetaFiles
	Enable the creation of request level WLM enclaves	578	WLMSN

Directive syntax guidelines

Each non-blank, non-comment record in the configuration file must contain one corresponding value, optionally followed by a comment. These records are not column-dependent and values are separated by one or more blanks. A directive and its values must be contained on a single line; continuation to the next line is not allowed.

The beginning of a comment is indicated by a pound sign (#). All characters from the pound sign to the end of the line are ignored. If either a pound sign or a blank needs to be specified for a directive, use the backslash (\) as an escape character before the pound sign or blank. For example, if \# is found on a line, the server interprets this as a pound sign character, not the beginning of a comment, and continues reading characters. Also, if \ is found on a line, the server interprets this as a blank, not a value delimiter, and continues reading characters to build the value.

Specifying *request-template* values

The value you specify for *request-template* is used to match an incoming URI to see if the directive will be used for the request. You can use the asterisk (*) as a wildcard character in templates. For all other characters, matching is done on a character-by-character and case-sensitive basis. For example, an incoming URI of `/abc/` or `/ABC` will not match a *request-template* of `/abc`.

Specifying a positive or negative string

Several configuration directives allow you to specify a positive or negative string.

For positive string, you can enter:
- Yes
- On
- OK

- Enable

For negative string, you can enter:
- No
- Off
- None
- Disable

Specifying time

Several configuration directives allow you to specify an amount of time. You can specify any combination of:

hh
: hours

hh:mm
: hours and minutes

hh:mm:ss
: hours, minutes, and seconds

n years
: number of 365-day years

n months
: number of 30-day months

n weeks
: number of 7-day weeks

n days
: number of 24-hour days

n hours
: number of 60-minute hours

n minutes
: number of 60-second minutes

n seconds
: number of seconds

All of your entries will be converted to seconds and added together.

Using an asterisk (*) as a wildcard character

Some directives contain templates for requests, path names, or host names (the *value* or *request-template* field). Except where otherwise indicated, you can use the asterisk (*) as a wildcard character in templates. For the template to be matched, an asterisk can be replaced by any character string or single character.

Using an at sign (@) in user IDs

If you specify a user ID containing an at sign (@) you must put double quotes around the user ID, for example:

```
"myid@rtp"
```

Using a blank or number sign in templates

The Web server interprets blanks as delimiters and a number sign (#) as the beginning of a comment that should be ignored. For example, a number sign precedes all comment lines in the Web server configuration file.

If you want to use a blank or number sign in templates, you must include a backslash (\) before the blank or number sign.

Access control - Set up access control for the server

Use the directives described in this section to control access to your server's resources.

Access control directives

You link protection setups to groups of files based on the requests that are used to access those files. Use the DefProt and Protect directives to define the requests you want to protect.

You can define the actual protection setup in a separate protection statement or directly in the configuration file. Within the configuration file, you can define and label a protection setup using the Protection directive. You can also define a protection setup directly on a DefProt or Protect directive.

This section also describes the subdirectives that define a protection setup.

DefProt - Specify default protection setup for requests that match a template

Use this directive to associate a default protection setup with requests that match a template.

Attention: For protection to work properly, you must put your DefProt and Protect directives in the configuration file before any Pass or Exec directives that match the template used on DefProt or Protect directives.

The general format of the directive follows:

```
DefProt request-template setup
[user_name [ServerIP-address or host_name]]
```

Note: You must type the directive on one line, even though the directive format is shown here on more than one line.

The format of the directive varies based on what you code for *setup*. *Setup* can be a protection setup label, a file with a full path, or in-line protection subdirectives. The format of the Defprot directive for each of these setup parameters follows:

```
DefProt request-template protection_setup_label
[user_name [ServerIP-address or host_name]]

DefProt request-template /full_path/file
[user_name [ServerIP-address or host_name]]

DefProt request-template
[user_name [ServerIP-address or host_name]] {
         in-line protection subdirectives
}
```

Note: The protection setup defined on the DefProt directive is the default protection for the Protect directive. You must have a Protect directive with the same template as the DefProt directive in order for the DefProt directive to be active. The template must be the only parameter on the Protect directive.

Valid DefProt parameter combinations follow:

```
DefProt request-template protection_setup_label
DefProt request-template protection_setup_label user_name
DefProt request-template protection_setup_label user_name ServerIP-address
DefProt request-template protection_setup_label user_name host_name
DefProt request-template protection_setup_label ServerIP-address
DefProt request-template protection_setup_label host_name
DefProt request-template /full_path/file
DefProt request-template /full_path/file user_name
DefProt request-template /full_path/file user_name ServerIP-address
DefProt request-template /full_path/file user_name host_name
DefProt request-template /full_path/file ServerIP-address
DefProt request-template /full path/file host_name
```

```
DefProt request-template user_name {
 < in-line protection subdirectives >
}
DEFPROT request-template user_name ServerIP-address {
 < in-line protection subdirectives >
}
DEFPROT request-template user_name host_name {
 < in-line protection subdirectives >
}
 DEFPROT request-template ServerIP-address {
 < in-line protection subdirectives >
}
DEFPROT request-template host_name {
 < in-line protection subdirectives >
}
DEFPROT request-template {
 < in-line protection subdirectives >
}
```

request-template

A template for requests that you want to associate with a default protection setup. The server compares incoming client requests to the template and associates a protection setup if there is a match.

The DefProt directive requires a request-template.

Protection is not actually activated for requests matching the template unless the request also matches a template on a subsequent Protect directive. See the description of the Protect directive for an explanation of how it is used with DefProt.

setup

The default protection setup you want to associate with requests that match *request-template*.

The DefProt directive requires one of the following three *setup* parameters. They are mutually exclusive.

- */full_path/file*: A full path and file name identifying a separate file that contains the protection subdirectives.
- *Protection_setup_label*: A protection setup label name that matches a name defined earlier on a Protection directive. The Protection directive contains the protection subdirectives.
- *In-line protection subdirectives*: The actual protection subdirectives. The subdirectives must be enclosed in braces {}. The left brace character must be the last character on the same line as the DefProt directive. Each subdirective follows on its own line. The right brace character must be on its own line following the last subdirective line. You cannot put any comment lines between the braces. See "Protection subdirectives" on page 444 for descriptions of the protection subdirectives.

user_name

The access control user to which the server should change when serving the request. This allows z/OS UNIX System Services file protection to restrict access. This parameter is optional. The *use_name* will be used for controlling access to MVS resources and must include an z/OS UNIX segment containing the UID and GID to be used for controlling access to HFS files. The *user_name* is only meaningful when the server is permitted to access BPX.SERVER for file requests and additionally to access BPX.DAEMON for CGI script requests. If *user_name* is not specified, the default access control user ID is used. See Chapter 1, "Planning for installation" on page 3 for information on access control user IDs and z/OS UNIX authorization environments.

If you use *user_name* in conjunction with either the server Internet Protocol (IP) address or the host name, *user_name* must precede the server IP address or the host name.

Server-IP-address **or** *host_name*
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. (For more information on using multiple IP addresses or virtual hosts, see Chapter 17, "Running your server with multiple IP addresses or virtual hosts" on page 309.) The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

The server's host name and IP address are mutually exclusive. You must fully qualify whichever one you use. You cannot specify a wild card (*). An example of a host name is hostA.bcd.com. An example of a server IP address is 204.146.167.72.

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

Examples:

```
DefProt /secret/*  /server/protect/setup1.acc   webname
```

The above example identifies a separate file that contains the protection subdirectives and a user ID for controlling access. For information about access control user IDs, see "Access control with z/OS RACF or other security products" on page 8.

```
DefProt /secret/*  SECRET-PROT  webname
```

The above example uses a label name to point to the protection subdirectives. The label name must match a label name on a Protection directive. The Protection directive must come before the DefProt directive. The access control user ID, webname, overrides any specified within the SECRET-PROT protection subdirective.

```
DefProt   /mysecret/* {
   AuthType Basic
   ServerID restricted
   PasswdFile /docs/etc/WWW/restrict.password
   GroupFile /docs/etc/WWW/restrict.group
   GetMask authors
   PutMask authors
}
```

The above example includes the protection subdirectives as part of the DefProt directive.

```
DefProt   /secret/*   CustomerA-PROT  webname  9.67.106.79
DefProt   /secret/*   CustomerB-PROT  webname  9.83.100.45
```

The above examples use the optional IP address parameter. If your server receives requests that begin with /secret/, it associates a different default protection setup with the request based on the IP address of the network connection the request comes in on. For requests coming in on 9.67.106.79, the server associates the request with default protection defined on a Protection directive with a label of CustomerA-PROT. For requests coming in on 9.83.100.45, the server associates the request with default protection defined on a Protection directive with a label of CustomerB-PROT.

```
DefProt   /secret/*   CustomerA-PROT  webname  hostA.bcd.com
DefProt   /secret/*   CustomerB-PROT  webname  hostB.bcd.com
```

The above examples use the optional *hostname* parameter. If your server receives requests that begin with /secret/, it associates a different default protection setup with the request based on the host name in the URL. For requests coming in for hostA, the server associates the request with default protection defined on a Protection directive with a label of CustomerA-PROT. For requests coming in on hostB, the server associates the request with default protection defined on a Protection directive with a label of CustomerB-PROT.

Program default setting

None

Initial configuration file setting

None

Protect - Activate protection setup for requests that match a template

Use this directive to activate protection setup rules for requests that match a template.

Attention: For protection to work properly, put your DefProt and Protect directives before any Pass or Exec directives in your configuration file.

Note: The server compares an incoming client request to the template on the protect directive. If the server finds a match between the incoming request and the template, the server continues to check subsequent Protect directives for additional matches. The server uses the last matching Protect directive for activating protection.So put your most restrictive Protect directives last.

The general format of the directive follows:

```
Protect request-template [setup]
[user_name [ServerIP-address or host_name]]
```

Note: You must type the directive on one line, even though the directive format is shown here on more than one line.

The format of the directive varies, based on what you code for *setup*. You can use *setup* as a protection setup label, a file with a full path, or in-line protection subdirectives. The format of the Protect directive for each of these setup parameters follows:

```
Protect request-template [protection_setup_label]
[user_name [ServerIP-address or host_name]]

Protect request-template [/full_path/file]
[user_name [ServerIP-address or host_name]]

Protect request-template
[user_name [ServerIP-address or host_name]] [{
         in-line protection subdirectives
}]
```

Valid Protect parameter combinations follow:

Note: If you specify a Protect directive with only the request-template parameter, specify a corresponding DefProt directive with the same request template.

Access control directives

```
Protect request-template
Protect request-template protection_setup_label
Protect request-template protection_setup_label user_name
Protect request-template protection_setup_label user_name ServerIP-address
Protect request-template protection_setup_label user_name host_name
Protect request-template protection_setup_label ServerIP-address
Protect request-template protection_setup_label host_name
Protect request-template /full_path/file
Protect request-template /full_path/file user_name
Protect request-template /full_path/file user_name ServerIP-address
Protect request-template /full_path/file user_name host_name
Protect request-template /full_path/file ServerIP-address
Protect request-template /full path/file host_name
Protect request-template user_name {
 < in-line protection subdirectives >
}
Protect request-template user_name ServerIP-address {
 < in-line protection subdirectives >
}
Protect request-template user_name host_name {
 < in-line protection subdirectives >
}
 Protect request-template ServerIP-address {
 < in-line protection subdirectives >
}
Protect request-template host_name {
 < in-line protection subdirectives >
}
Protect request-template {
 < in-line protection subdirectives >
}
```

request-template
A template for requests that you want to activate protection for. The server compares incoming client requests to the template and activates protection if there is a match.

Note: Use caution when specifying this value to ensure that resources are protected. When the Web server compares an incoming request to a *request-template*, it matches on a character-by-character and case-sensitive basis, not an implied resource basis.

For example, a Protect directive with a *request-template* of `/*.conf` will not match an incoming URI of `/server.conf/` or `/SERVER.CONF`, resulting in the resource not being protected by this Protect directive. In this example, even though the incoming URI `/server.conf/` would match on a file name and case basis, the match fails because a forward slash follows the file name.

setup
Use this parameter to identify the protection setup you want to activate for requests that match *request-template*. This parameter is optional. If it is omitted, the protection setup is defined by the most recent DefProt directive that contains a matching template.

The Protect directive requires one of the following three setup parameters. These parameters are mutually exclusive. See "Protection subdirectives" on page 444 for descriptions of the protection subdirectives.

- */full_path/file*: A full path and file name identifying a separate file that contains the protection subdirectives.

- *Protection_setup_label*: A protection setup label name that matches a name defined earlier on a Protection directive. The Protection directive contains the protection subdirectives.
- *In-line protection subdirectives*: The actual protection subdirectives. The subdirectives must be enclosed in braces {}. The left brace character must be the last character on the same line as the Protect directive. Each subdirective follows on its own line. The right brace character must be on its own line following the last subdirective line. You cannot put any comment lines between the braces.

user_name
The access control user to which the server should change when serving the request. This allows z/OS UNIX file protection to restrict access. This parameter is optional, but if you want to use it you must also use the *setup* parameter. The *username* will be used for controlling access to MVS resources and must include an z/OS UNIX segment containing the UID and GID to be used for controlling access to HFS files. The *user_name* is only meaningful when the server is permitted to access BPX.SERVER for file requests and additionally to access BPX.DAEMON for CGI script requests. If *user_name* is not specified, the default access control user ID is used. See Chapter 1, "Planning for installation" on page 3 for information on user IDs and z/OS UNIX authorization environments.

If you use *user_name* in conjunction with either the server Internet Protocol (IP) address or the host name, *user_name* must precede the server IP address or the host name.

Note: *user_name* is inherited from a DefProt directive to a Protect directive only when the protection setup is also inherited. If you use the *setup* parameter without the *user_name* parameter on a Protect directive, *user_name* defaults to the access control user ID regardless of any previous matching DefProt directives. Because of this default, the server will not run with the wrong user name for a given protection setup.

Server-IP-address **or** *host_name*
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. (For more information on using multiple IP addresses or virtual hosts, see Chapter 17, "Running your server with multiple IP addresses or virtual hosts" on page 309.) The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

The server host name and IP address are mutually exclusive. You must fully qualify whichever one you use. You cannot specify a wild card (*). An example of a host name is `hostA.bcd.com`. An example of a server IP address is 204.146.167.72.

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

Examples:

```
UserID  anybody
Protection BUS-PROT {
   UserID    busybody
   AuthType Basic
   ServerID restricted
   PasswdFile  /docs/WWW/restrict.pwd
```

```
      GroupFile   /docs/WWW/restrict.grp
      GetMask authors
      PutMask authors
   }
   DefProt /secret/*  /server/protect/setup1.acc   webname
   Protect /secret/scoop/*
   Protect /secret/business/*   BUS-PROT
   Protect  /verysecret/*  {
      AuthType Basic
      ServerID restricted
      PasswdFile /docs/WWW/restrict.pwd
      GroupFile  /docs/WWW/restrict.grp
      GetMask topbrass
      PutMask topbrass
   }
  Pass   /secret/scoop/*       /WWW/restricted/*
  Pass   /secret/business/*    /WWW/confidential/*
  Pass   /verysecret/*          /WWW/verysecret/*
```

In the above example, the server would activate protection as follows:

- Requests that start with /secret/scoop/ activate protection. The protection setup is defined in the /server/protect/setup1.acc protection setup file. Since the Protect directive does not specify a protection setup, the protection setup on the previously matching DefProt directive is used. Also, the server changes to the z/OS UNIX user of webname as defined on the DefProt directive.
- Requests beginning with /secret/business/ activate protection. The protection setup is defined on the Protection directive that has a label of BUS-PROT. Also, the server changes to the z/OS UNIX user of busybody as defined in the BUS-PROT protection setup.
- Requests beginning with /verysecret/ activate protection. The protection setup is included directly on the Protect directive. The z/OS UNIX user defaults to ANYBODY.

  Note: The user ID ANYBODY must be defined and the server must have permission to use it as a surrogate.

```
   Protect   /secret/*   CustomerA-PROT  webname  9.67.106.79
   Protect   /secret/*   CustomerB-PROT  webname  9.83.100.45
   Protect  /verysecret/*  webname 9.67.106.79 {
      AuthType Basic
      ServerID restricted
      PasswdFile /docs/WWW/customer-A.pwd
      GroupFile  /docs/WWW/customer-A.grp
      GetMask A-brass
      PutMask A-brass
         }
  Protect  /verysecret/*  webname   9.83.100.45   {
      AuthType Basic
      ServerID restricted
      PasswdFile /docs/WWW/customer-B.pwd
      GroupFile  /docs/WWW/customer-B.grp
      GetMask B-brass
      PutMask B-brass
         }
```

The above examples use IP addresses. If your server receives requests that begin with /secret/ or /verysecret/, it activates a different protection setup for the request based on the IP address of the network connection the request comes in on.

For /secret/ requests coming in on 9.67.106.79, the server activates the protection setup defined on a Protection directive with a label of CustomerA-PROT. For

/verysecret/ requests coming in on 9.67.106.79, the server activates the protection setup defined inline on the Protect directive for /verysecret/.

For /secret/ requests coming in on 9.83.100.45, the server activates the protection setup defined on a Protection directive with a label of CustomerB-PROT. For /verysecret/ requests coming in on 9.83.100.45, the server activates the protection setup defined inline on the Protect directive for /verysecret/.

```
Protect   /secret/*   CustomerA-PROT  webname  hostA.bcd.com
Protect   /secret/*   CustomerB-PROT  webname  hostB.bck.com
Protect  /verysecret/*  webname  hostA.bcd.com   {
   AuthType Basic
   ServerID restricted
   PasswdFile /docs/WWW/customer-A.pwd
   GroupFile  /docs/WWW/customer-A.grp
   GetMask A-brass
   PutMask A-brass
}
 Protect  /verysecret/*  webname hostB.bcd.com   {
   AuthType Basic
   ServerID restricted
   PasswdFile /docs/WWW/customer-B.pwd
   GroupFile  /docs/WWW/customer-B.grp
   GetMask B-brass
   PutMask B-brass
}
```

The above examples use virtual hosts. If your server receives requests that begin with /secret/ or /verysecret/, it activates a different protection setup for the request based on the host name in the URL.

For /secret/ requests coming in for hostA.bcd.com, the server activates the protection setup defined on a Protection directive with a label of CustomerA-PROT. For /verysecret/ requests coming in for hostA.bcd.com, the server activates the protection setup defined inline on the Protect directive for /verysecret/.

For /secret/ requests coming in for hostB.bcd.com, the server activates the protection setup defined on a Protection directive with a label of CustomerB-PROT. For /verysecret/ requests coming in for hostB.bcd.com, the server activates the protection setup defined inline on the Protect directive for /verysecret/.

Initial configuration file setting

Protection is provided for the Configuration and Administration forms by a Protect directive with a request template of /admin-bin/*.

Program default setting

None.

Protection - Define a named protection setup within the configuration file

Use this directive to define a protection setup within the configuration file. You give the protection setup a name and define the type of protection using protection subdirectives.

Note: In the configuration file, you must place Protection directives before any DefProt or Protect directives that point to them.

The format of the directive is:

```
Protection label-name  {
  subdirective  value
  subdirective  value
  .
  .
  .
}
```

label-name
The name you want to associate with this protection setup. The name can then be used by subsequent DefProt and Protect directives to point to this protection setup.

subdirective value
Put a protection subdirective and its value on each line between the left brace and the right brace. You cannot put any comment lines between the braces.

See "Protection subdirectives" for descriptions of the protection subdirectives.

Example

```
Protection NAME-ME   {
   AuthType Basic
   ServerID restricted
   PasswdFile  /WWW/password.pwd
   GroupFile   /WWW/group.grp
   GetMask groupname
   PutMask groupname
}
```

Initial configuration file setting

```
Protection IMW_Admin {
    ServerId     IMWEBSRV_Administration
    AuthType     Basic
    PasswdFile   %%SAF%%
    Mask         WEBADM, WEBADM
}
```

Program default setting

None

Protection subdirectives

This section includes descriptions of each of the protection subdirectives that can be used in a protection setup. Subdirectives are listed in alphabetical order but do not need to be in any particular order in your configuration file.

Protection setups can either be in separate files or within the configuration file as part of DefProt, Protect, or Protection directives. See the previous descriptions of the DefProt, Protect, and Protection directives for examples of using protection setups.

Note:
If you specify special characters such as a blank, tab, colon, comma, left parenthesis, right parenthesis, at sign, exclamation mark, or right brace within a Protection subdirective, enclose the subdirective value in double quotes, for example:

```
GroupFile "%%LDAP:SERVER1, SERVER2%%"
```

ACLOverride - Specify that ACL files override protection setups

Use this subdirective with a value of `On` if you want Access Control List files (ACL) to override the masks specified in the protection setup. If a directory being protected by the protection setup has an ACL file, the mask subdirectives in the protection setup are ignored. (The mask subdirectives are DeleteMask, GetMask, Mask, PostMask, and PutMask.)

Example:

```
ACLOverride On
```

AuthType - Specify authentication type

Use this subdirective when limiting access based on user names and passwords. Specify the type of authentication to use when the client sends a password to the server. With basic authentication (`AuthType Basic`), passwords are sent to the server as plain text. They are encoded, but not encrypted.

Example:

```
AuthType Basic
```

DeleteMask - Specify the user names, groups, and addresses allowed to delete files

Use this subdirective to specify user names, groups, and addresses templates authorized to make DELETE requests to a protected directory.

Example:

```
DeleteMask authors,(niceguy,goodie)@96.96.3.1,128.141.*.*
```

GetMask - Specify the user names, groups, and addresses allowed to get files

Use this subdirective to specify user names, groups, and address templates authorized to make GET requests to a protected directory.

Example:

```
GetMask authors,(niceguy,goodie)@96.96.3.1,128.141.*.*
```

GroupFile - Specify the location of the associated group file

Use this subdirective to specify the path and file name of the server group file that you want this protection setup to use. The groups defined within the server group file can then be used by:

- Any mask subdirectives that are part of the protection setup. (The mask subdirectives are DeleteMask, GetMask, Mask, PostMask, and PutMask.)
- Any ACL file on a directory that is protected by the protection setup.

For more information on server group files, see “Group files” on page 172.

Notes:

1. If you specify %%LDAP%% on the GroupFile subdirective, the value must be in double quotes. If double quotes are not used, syntax errors may occur when using certain special characters such as a blank, tab, colon, comma, left parenthesis, right parenthesis, at sign, exclamation mark, and right brace.
2. The LDAPInfo directive must precede any %%LDAP%% references in the configuration file.
3. You can code %%LDAP%% on the GroupFile subdirective only if you code %%LDAP%% on the PasswdFile subdirective.

Example:

```
GroupFile /docs/etc/WWW/restrict.group
GroupFile "%%LDAP[:PrimaryLdapServer[, SecondaryLdapServer]]%%"
```

PrimaryLdapServer and SecondaryLdapServer match LDAP servers defined in LDAPInfo directives, for example:

```
GroupFile "%%LDAP:SERVER1, SERVER2%%"
```

The server first looks to the PrimaryLdapServer for group information. If there is no response, the request is made to the SecondaryLdapServer. The server with whom a connection is made will be used to look up the password information

Mask - Specify the user names, groups, and addresses allowed to make HTTP requests

Use this subdirective to specify user names, groups, and address templates authorized to make HTTP requests not covered by other mask subdirectives. See "Methods - Set method acceptance" on page 518 for descriptions of the HTTP methods supported by the server.

Example:

```
Mask authors,(niceguy,goodie)@96.96.3.1,128.141.*.*
```

Note: When you use the Mask directive, it is important that you keep in mind that Masks are case sensitive. The following is an example of how you would issue Mask protection on a user ID:

Example:

```
MASK WEBADM,webadm
```

PasswdFile - Specify the location of the associated password file

Use this subdirective when limiting access based on user names and passwords. Specify the path name of the password file that you want this protection setup to use, or that you want to use the SAF interface to your system security subsystem to validate user names and passwords.

Because some browsers, such as Netscape, cache the userid and password by security realm (ServerID), follow these guidelines when specifying ServerID and password files:

- Protection setups that use the same password file should use the same ServerId strings to avoid reprompting at the browser to cache the same information multiple times.

- Protection setups that use different password files should use different ServerId strings. This forces prompting once per realm and avoids thrashing the browser's user ID or password cache each time realms are switched.

Notes:

1. In the following example, note that `%%SAF%%` lets the server know that password verification is done with a SAF-controlled operating system such as RACF. If the access control UserID resolves to %%CLIENT%% (from either the UserID subdirective or the UserID directive), SAF will always be used to verify the UserID and password, in addition to the given password file.
2. If you specify `%%LDAP%%` on the PasswdFile subdirective, the value must be in double quotes. If double quotes are not used, syntax errors may occur when using certain special characters such as a blank, tab, colon, comma, left parenthesis, right parenthesis, at sign, exclamation mark, and right brace.
3. The LDAPInfo directive must precede any `%%LDAP%%` references in the configuration file.

Examples:

```
PasswdFile %%SAF%%
PasswdFile "%%LDAP[:PrimaryLdapServer[, SecondaryLdapServer]]%%"
PasswdFile /usr/lpp/internet/server_root/Admin/heroes.pwd
```

`PrimaryLdapServer` and `SecondaryLdapServer` match LDAP servers defined in LDAPInfo directives, for example:

```
PasswdFile "%%LDAP:SERVER1, SERVER2%%"
```

The server first looks to the `PrimaryLdapServer` for group information. If there is no response, the request is made to the `SecondaryLdapServer`. The server with whom a connection is made will be used to look up the password information

PostMask - Specify the user names, groups, and addresses allowed to post files

For a secure server, use this subdirective to specify users, groups, and address templates authorized to make POST requests to a protected directory.

Example:

```
PostMask authors,(niceguy,goodie)@96.96.3.1,128.141.*.*
```

PutMask - Specify the users names, groups, and addresses allowed to put files

Use this subdirective to specify users, groups, and address templates authorized to make PUT requests to a protected directory.

Example:

```
PutMask authors,(niceguy,goodie)@96.96.3.1,128.141.*.*
```

ServerID - Specify a name to associate with the password file

Use this subdirective when limiting access based on user names and passwords. Specify a name you want to associate with the password file being used. The name does not need to be a real machine name.

The name is used as an identifier to the requester. Since different protection setups can use different password files, having a name associated with the protection setup can help the client decide which password to send. Most clients display this name when prompting for a user name and password.

Because some browsers such as NetScape cache userid/password by security realm (ServerID) within host, follow these guidelines when specifying ServerID and password files:

- Protection setups that use the same password file should use the same ServerID.
- Protection setups that use different password files should use different ServerIDs.

Example:

```
ServerID restricted
```

UserID - Specify the Access Control user ID that the server should use

Use this subdirective to specify the access control user ID that the server should change to when serving a request. You can use this subdirective only in a protection setup within the configuration file, not in a separate protection setup file. If you do not specify the access control user ID, the default access control user ID is used.

Web servers frequently need to process requests from users that do not have user IDs on the system running the server. The server uses access control User IDs to access resources for these requests. The HTTP Server uses four types of access control user IDs. There are special user IDs, %%CLIENT%%, %%SERVER%%, %%CERTIF%%, and surrogate.

%%CLIENT%%
: The Web server requires that the requester have a local z/OS user ID and password. The requester's user ID is used to access the data. The user is prompted for a valid password.

%%SERVER%%
: The Web server uses its own user ID to access data.

Note: Be extremely cautious when using %%SERVER%%. If the Web server is running with a user ID that has superuser authority (UID of 0), requests can be served under superuser authority.

%%CERTIF%%
: The Web server treats SSL connection certificate data in a special way. When presented with an SSL session with client certificate data present, the Web server attempts to map the client certificate to a local MVS User ID and password. The request is treated as if %%CLIENT%% has been specified in the following situations:
  - The session is not an SSL session.
  - There is no certificate present or the certificate cannot be mapped.
  - The underlying support is not available.

  Note that SSLCLientAuth must be set on in order to get client certificate data.

Related information:

- "Access control with z/OS RACF or other security products" on page 8
- "Web server surrogate user IDs" on page 22

- The ability to change to a different z/OS UNIX identity is controlled on MVS. The server must be given surrogate authority for the specified user ID and might be permitted to access the BPX.SERVER and BPX.DAEMON facilities. For more information about the BPX facilities, see Chapter 1, "Planning for installation" on page 3 and the *z/OS UNIX System Services Planning* book.

Example:

```
UserID WWW
UserID %%SERVER%%
```

Use %%SERVER%% if you are sharing information on an external LDAP server. See Chapter 15, "Retrieving Lightweight Directory Access Protocol information" on page 297 for more information.

Initial configuration file setting:

```
UserId  %%CLIENT%%
```

Program default setting:

```
UserId  %%CLIENT%%
```

SSL client authentication subdirectives

If you implement SSL client authentication, the server requests the client's certificate when the client makes an **https** request. You can use certain SSL parameters as subdirectives in Protection setups. For more information, see "Creating protection setups for Secure Sockets Layer (SSL) client authentication" on page 168.

Basic - Specify required settings

Use the directives described in this section to control your server's basic configuration settings.

BindSpecific - Specify if the server binds to one or all IP addresses

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive on a multi-networking system to run a different Web server on each IP address. All the servers may listen on the same port.

If BindSpecific is set on, the server binds to the IP address specified in the HostName directive only, instead of binding to all local IP addresses.

Note: For OS/390 Release 5 and later, you must use the SHAREPORT option on the PORT statement in your TCP/IP profile when you are configuring the Web server to use virtual hosting with BindSpecific on and using the same port for multiple Web servers.

If BindSpecific is not specified, the server binds to the default HostName.

Example

```
BindSpecific On
```

Initial configuration file setting

```
BindSpecific Off
```

Program default setting

```
BindSpecific Off
```

Bounce — Specify the default start option for the sockets setting SO_REUSEADDR

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

By default, the Web server starts with the sockets setting SO_REUSEADDR on. This will avoid a potential delay while sockets used by a previous invocation of the Web server are closed. This is especially useful if you have automation software that recognizes the Web server termination and needs to start a replacement Web server as quickly as possible.

You must use this directive to set SO_REUSEADDR off (Bounce off). To set Bounce off, you must also omit the -B flag from your Web server startup command. The startup command will override the directive setting. See "httpd command" on page 393 for information on the startup command.

If bounce is set off, the Web server will retry the bind to its listen ports for up to two minutes to allow previously used sockets to close. If the Web server is started quickly after a previous Web server shutdown and Bounce is set off, the listen ports may not be available. This can result in a delay in the availability of the Web server.

Example

```
Bounce Off
```

Initial configuration file setting

```
None
```

Program default setting

```
Bounce On
```

DNS-Lookup - Specify whether you want to look up host names of clients

Use this directive to specify whether you want your server to look up the host name of requesting clients.

The value you use affects the following things about how your server works:

- The performance of the server. Using the default value of `Off` improves the performance and response time of the server because it does not use resources to perform the host name lookup.

- The information your server records about clients when writing to log files.
  `Off` - Clients identified by IP address
  `On` - Clients identified by host name
- Whether you can use host names on address templates in protection setups, server group files, and ACL files.
  `Off` - Cannot use host names on address templates; must use IP addresses
  `On` - Can use host names on address templates; cannot use IP addresses

Example

```
DNS-Lookup   On
```

Initial configuration file setting

```
DNS-Lookup   Off
```

Program default setting

```
DNS-Lookup   Off
```

HostName - Specify the fully qualified domain name or IP address for the server

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the domain name or an IP address returned to clients from document requests. If you specify a domain name, a domain name server must be able to resolve the name into an IP address. If you specify an IP address, the domain name server is not needed or accessed.

Important migration note: In previous releases, the HostName directive bound the server to only the IP address specified, instead of binding to all local IP addresses. Currently, you must also specify BindSpecific On if you want the server to bind to only the IP address specified for this directive.

Example

```
HostName name or IP address
```

Initial configuration file setting

None.

Program default setting

Hostname <host name default defined in DNS>

imbeds - Specify whether server-side includes will be dynamically imbedded

Use this directive to specify if you want server-side include processing to be performed for documents served from the file system, CGI programs, or both. Server-side include processing is done on documents with a content type of text/x-ssi-html. Optionally, you can specify that server-side include processing also be done for documents with a content type of text/html. For more information about content types, see "AddType - Specify the data type of files with particular suffixes" on page 523.

Basic directives

You can use server-side includes to dynamically insert information, such as the date, the size of a file, the last change date of a file, CGI or server-side include environment variables, or text documents into the document being returned. For more information on using server-side includes, see "Using server-side includes to insert information into CGI programs and HTML documents" on page 214.

Server-side include processing causes the server to search your documents for special commands each time they are served. This can affect the server's performance and slow down response time to clients.

The format of this directive is:

```
imbeds source [type]
```

source can be:

on Server-side include processing is done for documents from the file system and from CGI programs.

files Server-side include processing is only done for documents from the file system.

cgi Server-side include processing is only done for documents returned by CGI programs.

off Server-side include processing is not done for any documents.

The server checks the content type of each file it retrieves and the output of each CGI program it processes.

Server-side include processing is normally done only for documents having a content type of text/x-ssi-html. However, you can specify that documents with a content type of text/html be processed for server-side includes.

type can be:

SSIOnly
Server-side include processing is done for documents with a content type of text/x-ssi-html.

html Server-side include processing is done for documents with a content type of text/html and a content type of text/x-ssi-html.

Note: The server treats html, .html, and .htm as **html**. Anything else is treated as **SSIOnly**.

Initial configuration file setting

```
imbeds on SSIOnly
```

Program default setting

```
imbeds off SSIOnly
```

InstallPath - Specify an alternate directory installation path

Use this directive to specify an alternate installation directory path for the HTTP Server. Installing the server in an alternate directory path allows you to install a new release of the server without disrupting the server you currently have running. You can sufficiently test the new release of the server while keeping the previous release of the server in production.

The server is installed into a directory other than the default directory of /usr/lpp/internet and is executed from the alternate directory. It is recommended that you install the new release of the server in a nonstandard place on a system with the previous release in the standard place. When the HTTP Server is installed in an alternate directory, the default configuration files created are automatically tailored to the alternate installation location. When you finish testing, you should be able to unmount the new server from its original location and remount it over /usr/lpp/internet to move it from test to production. Your current configuration files (httpd.conf, httpd.envvars, ics_pics.conf,lgw_fcgi.conf, mvsds.conf, javelin.conf, and socks.conf) may not be compatible with the new level of the server. You need to manually modify the configuration files to the new path, use the configuration files from the previously installed server, or use the default configuration files shipped in the installpath/etc directory.

Example

```
InstallPath /service/usr/lpp/internet
```

Initial configuration file setting

```
InstallPath  /usr/lpp/internet
```

Program default setting

```
InstallPath  /usr/lpp/internet
```

NoLastMod - Specify whether LastModified HTTP headers are added to CGI program output

Use this directive to specify whether LastModified HTTP headers are added to CGI program output. The LastModified header affects whether or not browsers will cache the page. The format of the directive is:

```
NoLastMod On|Off
```

On
: LastModified headers are *not* added to CGI output if they are not already present.

Off
: LastModified headers are added to CGI output if they are not already present.

Example

```
NoLastMod On
```

Initial configuration file setting

```
None.
```

Program default setting

```
NoLastMod  Off
```

PidFile - Specify the location of the process ID file

Important:
If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see “Directives that require a stop and restart of the server” on page 425.

Use this directive to specify the full path and file name of the file that you want the server to write its process ID to when you start the Web server.

Example

```
PidFile /other/ibm_server.pid
```

Initial configuration file setting

PidFile /copy/usr/lpp/internet/server_root/httpd-pid

Program default setting

PidFile /copy/usr/lpp/internet/server_root/httpd-pid

Port - Specify the port on which you want the server to listen for requests

Important:
If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the port number the server should listen to for requests. The standard port number for HTTP is 80. Other port numbers less than 1024 are reserved for other TCP/IP applications and should not be used. Common ports used for proxy Web servers are 8080 and 8008.

When a port other than 80 is used, clients are required to include a specific port number on requests to the server. The port number is preceded by a colon and placed after the host name on the URL. For example, from the browser, the URL

```
http://www.turfco.com:8008/
```

requests the default welcome page from a host named www.turfco.com that is listening on port 8008.

You can use the -p option on the httpd command to override this setting when starting the server.

Example

```
Port 8080
```

Initial configuration file setting

```
Port 80
```

Program default setting

```
Port 80
```

Recovery — Customize ABEND recovery performed by the Web server

Use this directive to control the action the Web server will take when it detects that an ABEND has occurred in its address space. For planning considerations, see "ABEND recovery performed by the Web server" on page 7.

The format of this directive is:

```
Recovery type
```

type can be:

None The Web server does not perform ABEND recovery.

MsgOnly
The Web server does not perform ABEND recovery. The Web server issues messages, then terminates. Neither a CEEDUMP, nor an IEATDUMP is taken.

Msg/Dump
The Web server does not perform ABEND recovery. The Web server issues messages, takes a CEEDUMP, takes an IEATDUMP if the SysDumpName directive is set, then terminates.

Normal
The Web server attempts ABEND recovery. The Web server issues messages, takes a CEEDUMP, takes an IEATDUMP if the SysDumpName directive is set, then recovers if possible. If the Web server runs in Scalable Server mode, it does not attempt ABEND recovery for WLM queue servers.

Full The Web server attempts ABEND recovery. The Web server issues messages, takes a CEEDUMP, takes an IEATDUMP if the SysDumpName directive is set, then recovers if possible. If the Web server runs in Scalable Server mode, ABEND recovery is attempted for WLM queue servers.

For more information on error conditions, see the descriptions of messages IMW0085E and IMW0162E in Appendix F, “Messages” on page 603.

Initial configuration file setting

`None.`

Program default setting

`Recovery  Normal`

Notes:

1. The CEEDUMP is a formatted dump with limited detailed information. This information is usually sufficient to solve problems. For problems that require more research, the Web server provides the capability to create an IEATDUMP. For an explanation on how to create an IEATDUMP, see “SysDumpName - Specify the high-level qualifier of the IEATDUMP data set” on page 456.
2. The CEEDUMP and the IEATDUMP are taken at the time the problem is first encountered. For problems requiring more extensive research, an IBM customer service representative can request that you recreate the problem with a SYSMDUMP. The SYSMDUMP provides a much more detailed unformatted dump. For information on how to code the SYSMDUMP DD card in your job control language (JCL), see “IMWHTTPD program” on page 400.
3. There are instances when the Web server will not take a dump, even though you have specified `Normal` or `Full` on the Recovery directive. In these cases an IBM customer service representative can request that you code `None` on the Recovery directive in the httpd.conf file and code the appropriate Language Environment runtime options.

ServerRoot - Specify the current working directory of the server

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the current working directory of the server. By default, this directory is located in the install path specified by the InstallPath directive. This directive is relative to the InstallPath directive. Logging directives use this current working directory as the default root when relative path names are used.

Example

```
ServerRoot /usr/lpp/internet/server_root
```

Initial configuration file setting

```
ServerRoot /usr/lpp/internet/server_root
```

Note: PASS and EXEC rules may be independent of this directory.

ServerToken - Specify whether you want the server to send server identifying information

Use the ServerToken directive to control whether the Web server sends the server type and version to the client. You can send the Web server type and version in the following ways:

- The Server response header
- The comment portion of the Via header. Proxies use the Via header.
- The link on the default error pages. The link goes to the front page of the server.

Note: This directive affects the entire Web server. You cannot enable or disable it on a virtual host basis.

The format of the directive follows:

```
ServerToken On|Off
```

On Send the Web server type and version.

Off Do not send the Web server type and version.

Example

```
ServerToken Off
```

Initial configuration file setting

```
None
```

Program default setting

```
ServerToken On
```

SysDumpName - Specify the high-level qualifier of the IEATDUMP data set

Set the SysDumpName directive to take advantage of the IEATDUMP capability added to the abend Recovery directive. The IEATDUMP causes an unformatted

dump to be created in an abend recovery situation along with the formatted CEEDUMP that is normally created. The IEATDUMP is readable with the interactive problem control system (IPCS), a dump reading tool. The IEATDUMP provides much more information regarding the problem than the CEEDUMP at the time of the initial abend. If you do not provide this directive and an ABEND recovery situation exists, no IEATDUMP generates. The lack of an IEATDUMP can cause the IBM customer service representative to request that you recreate the problem with a SYSMDUMP.

The format of the directive follows:

```
SysDumpName hlq
```

hlq Provide a high-level qualifier for the data set that contains the IEATDUMP. The Web server substitutes the high-level qualifier in the following IEATDUMP data set name:

```
<hlq>.WEBSERVE.Dyymmdd.Thhmmss
```

You do not need a DD statement for this data set.

Example

```
SysDumpName MYHLQ
```

Initial configuration file setting

```
None
```

Program default setting

```
None
```

Notes:

1. The Recovery directive IEATDUMP feature requires that you turn on program control for the "*hlq*.MIGLIB" data set. See "Step 2. Turn on program control for DLLs" on page 24.
2. The SysDumpName directive works in conjunction with the following Recovery directive values:
   - Msg/Dump
   - Normal
   - Full

   Set one of these values on the Recovery directive to generate the IEATDUMP.

Userid - Specify the Default Access Control user ID

Use this directive to specify the user name the server changes to before accessing files. The ability to change to a different z/OS UNIX identity is controlled on MVS. The server must be given surrogate authority for any specified surrogate user ID and might need to be permitted to access the BPX.SERVER and BPX.DAEMON facilities. For more information about BPX facilities, see Chapter 1, "Planning for installation" on page 3 and *z/OS UNIX System Services Planning*.

The server does not normally serve data from its own WEBSRV user ID because of its level of authority. The server uses access control user IDs to access resources for requests. Every request must have an access control user ID using the Protect, DefProt, or UserId directive. Therefore, any pass rules that do not invoke a protect will use this user ID. The HTTP Server uses four types of access control user IDs. There are special user IDs, %%CLIENT%%, %%SERVER%%, %%CERTIF%%, and surrogate.

%%CLIENT%%
This is not really a user ID. Instead, it tells the server to require that the requester have a local z/OS user ID and password. The requester's user ID is used to access the data. The user is prompted for a valid password.

%%SERVER%%
This is not really a user ID. Instead, it tells the server to use its own user ID to access data.

Note: Be extremely cautious when using %%SERVER%%. If the Web server is running with a user ID that has superuser authority (UID of 0), requests can be served under superuser authority.

%%CERTIF%%
This is not really a user ID. Instead, it tells the server to treat SSL connection certificate data in a special way. The web server, when presented with an SSL session with client certificate data present, attempts to map the client certificate to a local MVS User ID and password. If the session isn't SSL, or there isn't a certificate present, or the underlying support is not available, or the certificate can't be mapped, then the request is treated as if %%CLIENT%% has been specified. Note that SSLCLientAuth must be ON in order to get clientcertificate data.

Related information:

- For more information on access control user IDs, see Chapter 1, "Planning for installation" on page 3.
- "Default protection scheme when coding %%CLIENT%% on UserID directive" on page 176

Example

```
UserId webmaster
```

Initial configuration file setting

```
UserId %%CLIENT%%
```

Codepages - Specify default codepage environment

Use the directives described in this section for codepage translation, primarily between ASCII and EBCDIC. Each directive specifies one codepage. The two directives together define a pair that must be supported by the iconv() service.

DefaultFsCp - Specify server codepage

Important:
If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the default file system code page on the server. This is the EBCDIC codepage for local text files and text streams from applications (gateway). For example, acceptable values are IBM-1047 (Open Systems Latin 1 codepage) and IBM-939 (Japanese (Latin) Extended codepage). For information

about the iconv(s) service, see the Code Page Conversion Table, "Code Set Conversion Utilities" chapter in the *C/MVS Programming Guide*. You might find the *C/C++ Library Reference* helpful also.

You can use the -fscp option on the httpd command to override the DefaultFsCp directive when you start the server.

Example

```
DefaultFsCp   IBM-939
```

Initial configuration file setting

None.

Program default setting

```
DefaultFsCp IBM-1047
```

DefaultNetCp - Specify codepage

Important:
If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the default network codepage. This is the ASCII codepage used for text bodies on the network. Some acceptable values are:
- ISO8859-1 (8bit single-byte coded graphic character sets)
- IBM-932C (Shift JIS)
- IBM-eucJC (Japanese version of Extended Unix Code)

For information about the iconv(s) service, see the Code Page Conversion Table, "Code Set Conversion Utilities" chapter in the *C/MVS Programming Guide*. You might find the *C/C++ Library Reference* helpful also.

You can use the -netcp option on the httpd command to override the DefaultNetCp directive when you start the server.

Example

```
DefaultNetCp   IBM-932C
```

Initial configuration file setting

None.

Program default setting

```
DefaultNetCP   ISO8859-1
```

Directories and Welcome Page - Set viewing options

Use the directives described in this section to control how your server responds to requests containing a directory name. You can have the server search the directory for a welcome file to return, or you can have the server generate a directory listing.

By default, the server first looks for a welcome file. If no welcome file is present, the server displays a directory listing. Configuration settings control how directory listings appear and the icons that the listings use.

The server provides a set of default icons to use for directory listings. You can replace these icons with others using some of the directives described in this section.

AddBlankIcon - Specify the icon URI used to align the heading of directory listings

Use this directive to specify an icon to use for aligning the heading on directory listings. This can either be a blank icon or another icon you want to appear on the headings of your directory listings. For proper alignment, the icon you use must be the same size as the other icons you are using on your directory listings. The format of the directive is:

```
AddBlankIcon   icon-URI   alternate-text
```

icon-URI

The URI for the icon. The URI is translated through the mapping directives. For the icon to be retrieved, the mapping directives must allow the URI to be passed.

If you are using the server as a proxy, you must specify a fully qualified URL pointing to your server.

alternate-text

The alternate text to use for the icon if the requesting browser is not displaying graphics.

Example

```
AddBlankIcon /icons/logo.gif  logo
```

Initial configuration file setting

```
AddBlankIcon  /icons/blank.gif
```

The default does not specify alternative text since the icon is blank.

AddDirIcon - Specify the icon URI for directories on directory listings

Use this directive to specify an icon for representing a directory on a directory listing. The format of the directive is:

```
AddDirIcon   icon-URI   alternate-text
```

icon-URI

The URI for the icon. The URI is translated through the mapping directives. For the icon to be retrieved, the mapping directives must allow the URI to be passed.

If you are using the server as a proxy, you must specify a fully qualified URL pointing to your server. You must map the URL to a local file and make sure that the mapping directives allow the URL to be passed.

alternate-text

The alternate text to use for the icon if the requesting browser is not displaying graphics.

Example

```
AddDirIcon  /icons/direct.gif  DIR
```

Initial configuration file setting

```
AddDirIcon  /icons/dir.gif     DIR
```

AddIcon - Bind an icon to a MIME content-type or encoding-type

Use this directive to specify icons for representing files with a specific MIME content-type or encoding-type. The server uses the icons on directory listings. The format of the directive is:

```
AddIcon   icon-URI alternate-text
type-template
```

icon-URI

The URI for the icon. The URI is translated through the mapping directives. For the icon to be retrieved, the mapping directives must allow the URI to be passed.

If you are using the server as a proxy, you must specify a fully qualified URL pointing to your server. You must map the URL to a local file and make sure that the mapping directives allow the URL to be passed.

alternate-text
The alternate text to use for the icon if the requesting browser is not displaying graphics.

type-template
Either a MIME content-type or encoding-type template. Content-type templates always contain a slash. Encoding-type templates never have a slash.

Example

```
AddIcon   /icons/movie.gif    video   video/*
```

Initial configuration file setting

```
AddIcon  /internet/server_root/icons/binary.gif    BIN  binary
AddIcon  /internet/server_root/icons/text.gif      TXT  text/*
AddIcon  /internet/server_root/icons/image.gif     IMG  image/*
AddIcon  /internet/server_root/icons/movie.gif     MOV  video/*
AddIcon  /internet/server_root/icons/sound.gif     AU   audio/*
AddIcon  /internet/server_root/icons/tar.gif       TAR  multipart/*tar
AddIcon  /internet/server_root/icons/compress.gif  CMP  x-compress x-gzip
```

AddParentIcon - Specify the icon URI for a parent directory on directory listings

Use this directive to specify an icon for representing a parent directory on a directory listing. The format of the directive is:

```
AddParentIcon   icon-URI   alternate-text
```

icon-URI

The URI for the icon. The URI is translated through the mapping directives. For the icon to be retrieved, the mapping directives must allow the URI to be passed.

If you are using the server as a proxy, you must specify a fully qualified URL pointing to your server. You must map the URL to a local file and make sure that the mapping directives allow the URL to be passed.

alternate-text
The alternate text to use for the icon if the requesting browser is not displaying graphics.

Example

```
AddParentIcon /icons/parent.gif  UP
```

Initial configuration file setting

```
AddParentIcon /icons/back.gif      UP
```

AddUnknownIcon - Specify the icon URI for unknown file types on directory listings

Use this directive to specify an icon for representing files with an unknown file type on a directory listing. The format of the directive is:

```
AddUnknownIcon   icon-URI   alternate-text
```

icon-URI

The URI for the icon. The URI is translated through the mapping directives. For the icon to be retrieved, the mapping directives must allow the URI to be passed.

If you are using the server as a proxy, you must specify a fully qualified URL pointing to your server. You must map the URL to a local file and make sure that the mapping directives allow the URL to be passed.

alternate-text
The alternate text to use for the icon if the requesting browser is not displaying graphics.

Example

```
AddUnknownIcon /icons/saywhat.gif  huh
```

Initial configuration file setting

```
AddUnknownIcon /icons/unknown.gif  ???
```

AlwaysWelcome - Specify if a welcome file is returned for all directory requests

Use this directive to specify if you want your server to always handle directory requests by first searching the directory for a welcome file.

The default value is `On`, which means that the server always searches the directory for a welcome file. The Welcome directive specifies the names of the files that the server recognizes as welcome files.

If you change the value to `Off`, the server first checks the last character of directory requests for the slash (/) character. If a directory request ends with a slash, the server searches the directory for a welcome file. If a directory request does not end with a slash, the server attempts to return a directory listing.

If the server does not find a welcome file, or AlwaysWelcome is set to Off and the directory request does not end in a slash, the DirAccess directive controls whether or not the server responds to the request with a directory listing.

Note: Setting AlwaysWelcome to Off does not affect requests that contain only your server name without a directory name. The server will always handle

these requests by looking in your document root directory for a welcome file. The server cannot generate a directory listing for the document root directory.

Example

```
AlwaysWelcome Off
```

Initial configuration file setting

```
AlwaysWelcome On
```

Program default setting

```
AlwaysWelcome On
```

DirAccess - Control directory listings

Use this directive to specify whether you want your server to return directory listings when requested. The values on the Welcome and AlwaysWelcome directives determine when a request is interpreted as a request for a directory listing.

The default value is `Off`, which means that the server does not return directory listings for all directories and subdirectories. If you want to control which directories and subdirectories the server can return directory listings for, use:

```
DirAccess   Selective
```

If you change the value to `On`, the server will return directory listings.

If you change the value to `Selective`, the server will return directory listings for any directory that contains a file named .www_browsable. The contents of the .www_browsable file are not important; the server only checks for its existence.

Examples:

```
DirAccess Off
DirAccess Selective
```

Initial configuration file setting

```
DirAccess Off
```

DirReadme - Control directory README files

Use this directive to specify if and where you want your server to display directory listing README files.

The default value is `Top`, which means that when the server returns a directory listing, it searches the directory for a file named README. If README is found, the server puts the contents of the file at the top of the directory listing.

If you change the value to `Bottom`, the server searches for a README file, but puts the contents at the bottom of the directory listing.

If you change the value to `Off`, the server does not search the directory for a README file.

Examples:

```
DirReadme Bottom
DirReadme Off
```

Initial configuration file setting

```
DirReadme Top
```

DirShowBrackets - Use brackets around alternative text on directory listings

Use this directive to specify whether you want the server to put brackets around alternative text on directory listings. The directives that specify directory listing icons also contain alternate text. The alternate text is used in place of an icon if the requesting browser is not displaying graphics.

Example

```
DirShowBrackets Off
```

Initial configuration file setting

```
DirShowBrackets On
```

DirShowBytes - Show byte count for small files on directory listings

Use this directive to specify whether directory listings should include the exact byte count for files smaller than 1 KB.

A value of `Off` means the directory listing shows a size of 1 KB for all files that are 1 KB or smaller.

Example

```
DirShowBytes On
```

Initial configuration file setting

```
DirShowBytes Off
```

DirShowCase - Use case when sorting files on directory listings

Use this directive to specify whether directory listings should distinguish between uppercase and lowercase letters when sorting file names.

A value of `On` means uppercase letters are placed after lowercase letters.

Example

```
DirShowCase Off
```

Initial configuration file setting

```
DirShowCase On
```

DirShowDate - Show date last modified on directory listings

Use this directive to specify whether directory listings should include the last modification date for each file.

Example

```
DirShowDate Off
```

Initial configuration file setting

```
DirShowDate On
```

DirShowDescription - Show descriptions for files on directory listings

Use this directive to specify whether directory listings should include descriptions for HTML files. The descriptions are taken from the files' HTML <title> tags.

Example

```
DirShowDescription Off
```

Initial configuration file setting

```
DirShowDescription On
```

DirShowGroup - Show the group ID of files on directory listings

Use this directive to specify whether directory listings should include the Group ID of the files.

Example

```
DirShowGroup On
```

Initial configuration file setting

None.

Program default setting

Off

DirShowHidden - Show hidden files on directory listings

Use this directive to specify whether directory listings should include any hidden files on the directory.

The server considers any file that has a name beginning with a period (.) to be a hidden file.

Example

```
DirShowHidden Off
```

Initial configuration file setting

```
DirShowHidden On
```

DirShowIcons - Show icons in directory listings

Use this directive to specify whether you want your server to include icons in directory listings. Icons can be used to provide a graphic representation of the content type of the files in the listing. The icons themselves are defined by the AddBlankIcon, AddDirIcon, AddIcon, AddParentIcon, and AddUnknownIcon directives.

Example

```
DirShowIcons Off
```

Initial configuration file setting

```
DirShowIcons On
```

DirShowMaxDescrLength - Set the maximum description length on directory listings

Use this directive to set the maximum number of characters to show in the description field on directory listings.

Example

```
DirShowMaxDescrLength 30
```

Initial configuration file setting

```
DirShowMaxDescrLength 25
```

DirShowMaxLength - Set the maximum length for file names on directory listings

Use this directive to set the maximum number of characters that will be used for file names on directory listings.

Example

```
DirShowMaxLength 30
```

Initial configuration file setting

```
DirShowMaxLength 25
```

DirShowMinLength - Set the minimum length for file names on directory listings

Use this directive to set the minimum number of characters that will always be reserved for file names on directory listings. Lengths of fully-qualified file names in the directory can exceed this number. However, file names cannot be longer than the number specified on the DirShowMaxLength directive.

Example

```
DirShowMinLength 10
```

Initial configuration file setting

```
DirShowMinLength 15
```

DirShowMode - Show file permissions on directory listings

Use this directive to specify whether directory listings should include the permission bits for each file.

Example

```
DirShowMode On
```

Initial configuration file setting

None.

Program default setting

```
DirShowMode Off
```

DirShowOwner - Show file owner on directory listings

Use this directive to specify whether directory listings should include the owner ID for each file.

Example

```
DirShowOwner On
```

Initial configuration file setting

None.

Program default setting

```
DirShowOwner Off
```

DirShowSize - Show file size on directory listings

Use this directive to specify whether directory listings should include the size of each file.

Example

```
DirShowSize Off
```

Initial configuration file setting

```
DirShowSize On
```

IconPath - Specify the path for the directory listing internal icons

Use this directive to specify the path where the icons you want to use on directory listings are stored. You can use this directive if you have a group of servers that you want to share the same set of icons.

Attention: This directive must be before any of the other icon directives (AddBlankIcon, AddDirIcon, AddParentIcon, AddUnknownIcon, and AddIcon).

Example

```
IconPath http://icon.server.com:8080/httpd-internal-icons/
```

In the above example, each request for a directory list icon generates a request to a server named icon.server.com.

Initial configuration file setting

```
None
```

Program default setting

```
IconPath  server_root/icons/*
```

Welcome - Specify names of welcome files

Use this directive to specify the name of a welcome file the server should look for to respond to requests that do not contain a specific file name. You can build a list of welcome files by putting multiple occurrences of this directive in the configuration file.

For requests that do not contain a file name or a directory name, the server always looks in the document root directory for a file that matches a name specified on a Welcome directive. If a match is found, the file is returned to the requester.

For requests that contain a directory name but not a file name, the AlwaysWelcome directive controls whether the server looks in the directory for a welcome file to return. By default, AlwaysWelcome is set to a value of `On`. This means the server

always looks in the requested directory for a file matching a name specified on a Welcome directive. If a match is found, the file is returned to the requester.

If the server finds more than one match between files in a directory and file names on Welcome directives, the order of the Welcome directives determines which file is returned. The server uses the Welcome directive closest to the top of the configuration file.

If the server does not find a welcome file in the directory, the DirAccess directive controls whether or not the server responds to the request with a directory listing.

The format of the Welcome directive is:

`Welcome` *`file-name`* `[`*`Server-IP-address`* `or` *`hostname`*`]`

file-name
: A file name you want to define as being a welcome file.

Server-IP-address **or** *hostname*
: If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. (For more information on using multiple IP addresses or virtual hosts, see Chapter 17, "Running your server with multiple IP addresses or virtual hosts" on page 309.) The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

  You can specify an IP address (for example, 204.146.167.72) or you can specify a host name (for example, hostA.bcd.com).

  This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URLs.

  A wildcard character cannot be specified for a server's IP address.

Examples:

```
Welcome letsgo.html
Welcome Welcome.html
```

The above example defines two welcome pages and assumes the AlwaysWelcome directive is set to its default of `On`. For requests that do not contain a file name, the server would try to return a welcome file from the directory specified on the request (or document root directory if the request does not specify a file name or a directory). The server would first look for a file named letsgo.html. If the directory does not have a letsgo.html file, the server would look for a file named Welcome.html.

```
Welcome CustomerA.html  9.67.106.79
Welcome CustomerB.html  9.83.100.45
```

Your server would look for different welcome files based on the IP address of the network connection the request comes in on. For requests coming in on 9.67.106.79 the server would look for welcome files named CustomerA.html. For requests coming in on 9.83.100.45, the server would look for welcome files named CustomerB.html. If the request comes in on a different IP address, the server looks for the default address.

```
Welcome CustomerA.html  hostA.bcd.com
Welcome CustomerB.html  hostB.bcd.com
```

Your server would look for different welcome files based on the host name in the URL. For requests coming in for hostA, the server would look for welcome files named CustomerA.html. For requests coming in for hostB, the server would look for welcome files named CustomerB.html. If the request comes in for a different host, the server looks for the default host name.

Initial configuration file setting

```
Welcome Welcome.html
Welcome welcome.html
Welcome index.html
Welcome Frntpage.html
```

Program default setting

```
Welcome.html
welcome.html
indexl.html
```

The above default values are shown in the order used by the default configuration.

User directories

Use the directive described in this section to control whether individual users of your server can have their own private Web documents.

UserDir - Enable users to have private Web documents

Use this directive to allow individual users of your server to have their own private Web documents. The user name must reference a defined z/OS UNIX user ID that includes a home directory definition. The value on the directive specifies the name of a subdirectory within each user's home directory. You must include the tilde (~) character before the user ID in the URL. For example, when the server receives a URL request that begins with / ~*userid*/, the server looks for the requested object in the subdirectory name specified on the UserDir directive of the user's home directory.

Example

```
UserDir public
```

In this example, if the server receives the request http://MyServer/ ~user129, the server looks in the user129/public directory and returns the appropriate welcome document (for example, index.html).

Default

None

Error messages - Customize Web server error messages

Use this directive to customize the messages your server sends to the requesting client when it encounters an error condition. For example, you can change a message to include more information about the cause of the problem and suggest possible solutions to fix it. For internal networks, you might provide a contact person for your users to call.

Each error condition is identified by a keyword. To decide which error messages you want to customize, first review the list of error conditions, their causes, and the default message that the server sends. Then, for each error message you want to change:

- Create an individual HTML file with the desired text.

- Add an **ErrorPage** directive to your configuration file that associates the error condition keyword with the HTML file you want to serve.

Note: The server does not parse your error files for imbeds, regardless of the file extensions or use of the Imbeds directive.

ErrorPage - Specify a customized message for a particular error condition

Use this directive to specify the name of a file that you want to send when the server encounters a particular error condition.

You can place this directive anywhere in the configuration file. When the error occurs, the file will be processed according to the mapping rules defined in your configuration file. Therefore, the file you want to send must be in a location that can be reached through the mapping rules as defined by the Fail, Map, NameTrans, Pass, Redirect, Service directives. At a minimum, you need a Pass directive that would allow the server to pass the error message file.

The format of this directive is:

```
ErrorPage keyword /path/filename.html
```

keyword
One of the keywords associated with an error condition. See "Error conditions, causes, and default messages" on page 471 for a list of keywords.

/path/filename.html
This is the fully qualified Web name of your error file, as viewed by a client on the Web.

Example

```
ErrorPage scriptstart /errors/html/scriptstart.html
```

In the above example, when a **scriptstart** condition is encountered, the server will send the scriptstart.html file found in the /errors/html directory to the client.

This file might contain the following HTML text:

```
<HTML>
<HEAD>
<TITLE>Message for SCRIPTSTART condition</TITLE>
</HEAD>
<BODY>
The CGI program could not be started.
<P>
<A HREF="mailto:admin@websvr.com">Notify the administrator</A>
of this problem.
</BODY>
</HTML>
```

If the directive that matches the above path in the server's configuration file is `PASS /* /wwwhome/*`, then the full path for this message file would be `/wwwhome/errors/html/scriptstart.html`.

Initial configuration file setting

None.

Program default setting

None.

Error conditions, causes, and default messages

The following list shows the HTTP status code and keyword for each error condition, followed by the probable cause, and the default message the server sends.For a list of valid return codes from MVSDS functions, see “Return codes from MVSDS functions” on page 601.

302 okredirect

Cause: The requested file is on another recognized server. The name of the server is sent back to the requesting client along with a message. The client can connect to the correct server or display the message that is sent.

Default message: Found.

400 badrequest

Cause: Either there is a network problem, such as a time-out, or the request was indecipherable.

Default message: Invalid request - completely unable to parse it.

400 badscript

Cause: The server could determine that the requested file was a CGI script but it could not process it; the request was invalid in some way.

Default message: The script execution request is not valid.

400 connectfail

Cause: On a tunneled request, the server could not connect to the requested partner on the requested port.

Default message: Host not found or not responding.

400 nopartner

Cause: On a tunneled request, the server could not connect to the requested hostname due to bad syntax or an unknown host.

Default message: Host not found or not responding.

400 proxyfail

Cause: The client is trying to use the server as a proxy, and although this is allowed, it did not work. Possibly the destination server doesn’t exist or is busy.

Default message: Proxy load failed.

400 unknownmethod

Cause: The request did not include a recognized method, such as GET, POST, PUT, or DELETE.

Default message: The request is not valid or not recognized.

401 notauthorized

Cause: The request requires a user ID and password. Either the user ID and password sent by the client are not valid for this request or the client did not send a user ID and password.

Default message: Not Authorized. Authentication failed.

401 notmember

Cause: The requested file has a protection rule listing valid user IDs and passwords and the user ID of the requesting client is not included in that list.

Default message: Not authorized to access the document.

Error message customization directives

401 pwchanged

Cause: The user ID has been changed to use the new password you entered. Enter the new password again to correct your browser's password cache.

Default message: Password changed. Enter *new_password* to continue.

401 pwexpired

Cause: The password for the MVS user ID has expired.

Default message: Access denied - password expired. Enter *old_password/new_password/new_password* to change your password.

401 pwnewinv

Cause: The password you entered did not meet the password format defined in the installation rules.

Default message: New password format not valid, try again. Enter *old_password/new_password/new_password* to change your password.

401 pwnewneq

Cause: The two passwords you entered for *new_password* do not match.

Default message: New passwords are not equal, try again. Enter *old_password/new_password/new_password* to change your password.

403 badredirect

Cause: The server is trying to redirect the request and the **Redirect** directive is invalid (possibly missing a destination) or contains a loop.

Default message: The redirection in the configuration file is not valid.

403 baduser

Cause: The client requested a user's home directory that does not exist.

Default message: The user directory is not valid.

403 byrule

Cause: Either the file requested is specifically blocked by a **Fail** directive or it does not match any of the files that are allowed to be accessed according to other request mapping directives.

Default message: Forbidden by rule.

403 dirbrowse

Cause: The client specified a directory (rather than a file name) in the URL that does not have a welcome page and the administrator has turned off directory browsing (either for this directory or for the entire server).

Default message: Directory browsing failed - access forbidden.

403 dotdot

Cause: The client request contains an instruction (/../) to navigate above the document directory root and this is not allowed.

Default message: Forbidden - URL containing .. forbidden (don't try to break in).

403 ipmask

Cause: The file requested has a protection rule that includes a list of valid IP addresses and the client's address is not included in the list.

Default message: Server will not serve to your IP address.

403 ipmaskproxy

Cause: The client is trying to use the server as a proxy and the client is not included in the list of host names or IP addresses that are allowed to do so.

Default message: Proxy server will not serve to your IP address (at least with this HTTP method).

403 methoddisabled

Cause: The client requested a method (such as GET, POST, PUT, DELETE) that is specifically not allowed by the Disable directive.

Default message: Method *method* is disabled on this server.

403 noacl

Cause: The directory has a protection rule but does not have an Access Control List (ACL) defined and the protection setup does not have a GetMask subdirective. The administrator needs to remove the protection rule or add an ACL.

Default message: Access to this file is not allowed 'no ACL file'.

403 noentry

Cause: The directory is protected by an Access Control List (ACL) and the user is not included in the ACL.

Default message: Access to this file is not allowed (no ACL entry).

403 notallowed

Cause: The requested file was found but the server's protection setup prevented access. This is commonly generated for URLs that point to CGI programs.

Default message: The PUT and DELETE methods must be specified in the server's protection setup.

403 openfailed

Cause: After passing the protection rules, the server determined that the client should have read access to the file but the operating system will not allow the server to access it. Possibly the user ID running the server does not have read permission to the file it is trying to serve or the file system may be encountering problems.

Default message: Can't browse selected file.

403 setuperror

Cause: The directory has an Access Control List (ACL) defined but does not have a protection rule. The administrator needs to add a protection rule or remove the ACL.

Default message: Server protection setup error occurred. Probably, the protection setup file was not found or it contained a syntax error.

404 multifail

Cause: The requested file could not be found on the server. The server tried to match the file name exactly as specified and with every known file extension appended.

Default message: The file was not found, even after searching on any extensions to the file name.

Error message customization directives

404 noapplenv

Cause: The request matched an ApplEnv definition, but the transfer of work to a queue server was not successful. The request may be processed in the queue manager.

Default message: Application Environment currently not available.

406 notacceptable

Cause: A request was submitted that matched one or more files found on the server but the accept headers sent with the request did not match exactly. For example, the accept language header asked for English files, but the matching file was French.

Default message: Not Acceptable - no file exists that matches the accept headers.

404 notfound

Cause: The requested file or directory cannot be served because it either does not exist or the client does not have permission to access it.

Default message: Not found. The file or directory does not exist or is read-protected.

407 proxypwchanged

Cause: The user ID has been changed to use the new password you entered. Enter the new password again to correct your brower's password cache.

Default message: Password changed. Enter *new_password* to continue.

407 proxypwexpired

Cause: The proxy password for the MVS user ID has expired. The password for the MVS proxy user ID has expired.

Default message: Access denied - password expired. Enter *old_password/new_password/new_password* to change your password.

407 proxynotauth

Cause: The proxy request requires a user ID and password. Either the user ID and password sent by the client are not valid for this request or the client did not send a user ID and password. Note that some Web browsers do not support the PROXY-AUTHENTICATE function.

Default message: Not authorized. Proxy-Authentication failed (or your browser does not support it).

407 proxynotmember

Cause: The proxy request has a protection rule listing valid user IDs and the user ID of the requesting client is not included in that list.

Default message: Not authorized for proxy access to the document.

407 proxypwnewinv

Cause: The password you entered did not meet the password format defined in the installation rules.

Default message: New password format not valid, try again. Enter *old_password/new_password/new_password* to change your password.

407 proxypwnewneq

Cause: The new password you entered is not correct. The two passwords you entered for *new_password* do not match.

Default message: New passwords are not equal, try again. Enter *old_password/new_password/new_password* to change your password.

412 preconfail

Cause: A precondition specified by the client on this request was not met. For example, this could result from an HTTP/1.1 request with a condition "if-not-modified-since *xxx*".

Default message: Precondition failed: could not match entity tags.

416 badrange

Cause: A PUT request either has an invalid content range header or it has incorrect information in the content range header for the file being processed. For example, the starting byte range of the associated content exceeds the existing file size. Note that a HTTP response code of 501 is returned if the content header cannot be parsed.

Default message: Invalid request - Content range is incorrect.

417 expectfailed

Cause: A request was submitted with an expect header but the server could not understand or honor the expectation sent in the header.

Default message: Expectation failed

500 addrspacedirty

Cause: The BPX.SERVER FACILITY or BPX.DAEMON FACILITY is defined and a program or DLL has been loaded into the server's address space that is not under PROGRAM CONTROL. The server's authority to check passwords and set access control user IDs has been temporarily revoked. You must stop the server, correct the problem, and start the server again. For information about controlling programs used with the server, seeChapter 1, "Planning for installation" on page 3.

Default message: Access denied - unauthorized program loaded.

500 setupsurrogate

Cause: A surrogate user ID is defined in the configuration file, but the server does not have permission to use this user ID as a surrogate. For information about creating surrogate user IDs, see Chapter 1, "Planning for installation" on page 3.

Default message: Access denied - surrogate user setup error.

500 scriptio

Cause: The client requested a CGI script; the server can find it and start it but cannot get it to process input or output. The script may contain invalid code.

Default message: Cannot read script output pipe.

500 scriptnotfound

Cause: The client requested a CGI script that cannot be found.

Default message: The script request is not valid; none of *<program>* and *<program>*.pp is executable.

500 scriptstart

Cause: The client requested a CGI script; the server can find it but cannot start it. The script may contain invalid code.

Default message: Starting the CGI program failed. Could not communicate with the CGI program.

500 systemerror
Cause: An internal MVS error occurred using SAF services. See trace table error information.

Default message: Access denied - system error using SAF.

501 noformat
Cause: The server has encountered an internal error and cannot interpret the format of the file it is trying to serve. The file may be corrupted or have an unknown or invalid file extension.

Default message: Sorry, can't convert from *mime-type-1* to *mime-type-2*.

503 serviceunavailable
Cause: A request thread is unavailable at the time of the request.

503 throttled
Cause: A proxy requested has been rejected because Web Traffic Express (WTE) throttling has been configured, and the server has determined that too many requests are currently being handled.

Default message: This server cannot accept any more requests right now. Please try again later.

Log401Error - Specify whether IMW0196I NOT AUTHENTICATED should be written to the error log

Use this directive to specify whether or not the Web server should write the informational message `IMW0196I NOT AUTHENTICATED` to the error log. The user ID and password are asked for each time a `401 NOT AUTHORIZED` error is encountered. If you set `Log401Error` to `YES`, the server places the informational message in the error log each time a user is prompted for a user ID and password. However, this could cause quite a few informational messages being added to the error log, resulting in a very large error log.

The value of this directive will have no effect on the 401 error counter for SMF and SNMP.

The format of this directive is:

```
Log401Error  Yes | No | Auth
```

Yes
The Web server will always write the informational message, `IMW0196I NOT AUTHENTICATED`, to the error log.

No
The Web server will never write the informational message, `IMW0196I NOT AUTHENTICATED`, to the error log

Auth
The Web server will write the informational message, `IMW0196I NOT AUTHENTICATED`, to the error log only if the request is accompanied by an authorization header and the header fails authorization.

Example

```
Log401Error Auth
```

Initial configuration file setting:

```
None
```

Program default setting:

`Log401Error Yes`

GWAPI - Specify GWAPI applications for processing

The Go Webserver Application Programming Interface (GWAPI) allows you to extend the HTTP Server's base functions with your own customized processing routines. Use the directives described in this section to have the server call the application functions in your program at various points in its request processing cycle.

For detailed information on writing the application functions and compiling your program, see Chapter 19, "Writing GWAPI programs" on page 345.

Except for Service and NameTrans, these directives can be in any order in the configuration file and you do not need to include every directive. If you do not have a customized application function for a particular step, just omit the corresponding directive. The normal processing for that step will execute by default.

The Service and NameTrans directives behave like the other mapping directives and are sensitive to their placement in the configuration file.

You can also have more than one configuration directive for a step. For example, you could include two NameTrans directives, each pointing to a different application function. When the server performs the name translation step, it will process your name translation functions in the order in which they appear within the configuration file.

Your application functions do not have to be executed for every request:

- By specifying a URI with some directives, you can indicate that you want the application function called only for URIs that match a certain pattern or mask.
- By specifying an authentication scheme with the Authentication directive, you can indicate that you want the application function called only for certain types of authentication.

Customize Web server process steps

The directives in this section are used to customize Web server process steps. For additional information, see "Server request process" on page 347.

ServerInit - Customize the initialization step

Use this directive to specify a customized application function you want the server to call during its initialization routines. This code will be executed before any client requests are read and whenever the server is restarted.

If you use this directive to preload frequently accessed data sets or PDS members, see Appendix E, "GWAPI MVSDS DLL Service" on page 597 for more information.

The format of the directive is:

```
ServerInit /path/file:function_name
```

/path/file
: The fully qualified file name of your compiled program, including the extension.

function_name
The name you gave your application function within your program.

Example:

```
ServerInit   /ics/api/bin/icsext05.so:svr_init
```

Initial configuration file setting: None.

Program default setting: None.

WLMClassify - Customize the WLM pre-exit step

Use this directive to specify a customized application function you want the server to call during the WLM pre-exit step. When the Web server is running in Scalable Server mode, this code will be executed after a client request has been read but before the work is put on the work queue.

The format of the directive is:

```
WLMClassify    /path/file:function_name
```

/path/file
The fully qualified file name of your compiled DLL, including the extension.

function_name
The name you gave your application function within your program.

Example:

```
WLMClassify   /ics/api/bin/icsext05.so:wlm_classify
```

Initial configuration file setting: None.

Program default setting: None.

PreExit - Customize the pre-exit step

Use this directive to specify a customized application function you want the server to call during the pre-exit step. This code will be executed after a client request has been read but before any other processing occurs.

The format of the directive is:

```
PreExit    /path/file:function_name
```

/path/file
The fully qualified file name of your compiled DLL, including the extension.

function_name
The name you gave your application function within your program.

Example:

```
PreExit      /ics/api/bin/icsext05.so:pre_exit
```

Initial configuration file setting: None.

Program default setting: None.

NameTrans - Customize the Name Translation step

Use this directive to specify a customized application function you want the server to call during the Name Translation step. This code would supply the mechanism for translating the virtual path in the request to the physical path on the server, mapping URIs to specific objects.

Note: This is not a terminal mapping rule. The transformed URI still has to match one of the terminal mapping rule directives, such as Exec, Fail, Map, Pass, Redirect, and Service.

The format of the directive is:

```
NameTrans request-template /path/file:function_name
[Server-IP-address or hostname]
```

Note: The directive must be typed on one line, even though it is shown here on two lines.

request-template
A template for requests that further determine if your application function is called. The specification can include the protocol, domain and host, can be preceded by a slash (/), and can use an asterisk (*) as a wildcard. For example, `/front_page.html`, `http://www.ics.raleigh.ibm.com`, /pub*, /*, and * are all valid.

/path/file
The fully qualified file name of your compiled program, including the extension.

function_name
The name you gave your application function within your program.

Server-IP-address **or** *hostname*
If you are using multiple IP addresses or virtual hosts, determines if your application function will be called only for requests coming in on a specific IP address or for a specific host.

A wildcard character cannot be specified for a server's IP address.

Example:

```
NameTrans /index.html /api/bin/icsextpgm.so:trans_url
```

Initial configuration file setting: None.

Program default setting: None.

Authorization - Customize the Authorization step

Use this directive to specify a customized application function you want the server to call during the Authorization step. This code would verify that the requested object can be served to the client.

The format of the directive is:

```
Authorization request-template /path/file:function_name
[Server-IP_address or hostname]
```

Note: The directive must be typed on one line, even though it is shown here on two lines.

request-template
A template for requests that further determine if your application function is called. The specification can be preceded by a slash (/), and can use an asterisk (*) as a wildcard. For example, `/front_page.html`, /pub*, /*, and * are all valid.

/path/file
The fully qualified file name of your compiled program, including the extension.

function_name
The name you gave your application function within your program.

Server-IP_address **or** *hostname*
If you use multiple IP addresses or virtual hosts, this option determines if your application function will be called only for requests coming in on a specific IP address or for a specific host.

You cannot specify a wild card character for a server's IP address.

Example:

```
Authorization /index.html /api/bin/icsextpgm.so:auth_url
```

Initial configuration file setting: None.

Program default setting: None.

Authentication - Customize the Authentication step

Use this directive to specify a customized application function you want the server to call during the Authentication step. This code will be executed based on the authentication scheme. Currently, only **Basic** authentication is supported.

Note: Authentication is part of the authorization process; it only occurs when authorization is required.

The format of the directive is:

```
Authentication type  /path/file:function_name
```

type
Specifies an authentication scheme which further determine if your application function is called. Both an asterisk (*) and `Basic` are accepted values.

/path/file
The fully qualified file name of your compiled program, including the extension.

function_name
The name you gave your application function within your program.

Example:

```
Authentication BASIC /ics/api/bin/icsextpgm.so:basic_authentication
```

Initial configuration file setting: None.

Program default setting: None.

ObjectType - Customize the Object Type step

Use this directive to specify a customized application function you want the server to call during the Object Type step. This code would locate the requested object in the file system and identify its MIME type.

The format of the directive is:

```
ObjectType request-template /path/file:function_name
```

request-template
A template for requests that further determine if your application function is called. The specification can include the protocol, domain and host, can be

preceded by a slash (/), and can use an asterisk (*) as a wildcard. For example, `/front_page.html`, `http://www.ics.raleigh.ibm.com`, /pub*, /*, and * are all valid.

/path/file
The fully qualified file name of your compiled program, including the extension.

function_name
The name you gave your application function within your program.

Example:

```
ObjectType /index.html /api/bin/icsextpgm.so:obj_type
```

Initial configuration file setting: None.

Program default setting: None.

Service - Customize the Service step

Use this directive to specify a customized application function you want the server to call during the Service step. This code would service the client request. For example, it sends the file or runs the CGI program.

Once a request matches a template on a Service directive, the request is not compared to request templates on any subsequent directives.

There is no default for this directive. If the request matches a Service rule (an application function specified on a Service directive is executed) but it returns HTTP_NOACTION, the server will generate an error and the request will fail.

The format of the directive is:

```
 Service request-template
/path/file:function_name [Server-IP_address or hostname]
```

Note: The directive must be typed on one line, even though it is shown here on more than one line.

request-template
A template for requests that further determine if your application function is called. The specification can include the protocol, domain and host, can be preceded by a slash (/), and can use an asterisk (*) as a wildcard. For example, `/front_page.html`, `http://www.ics.raleigh.ibm.com`, /pub*, /*, and * are all valid.

/path/file
The fully qualified file name of your compiled program, including the extension.

function_name
The name you gave your application function within your program.

Server-IP_address **or** *hostname*
If you use multiple IP addresses or virtual hosts, this option determines if your application function will be called only for requests coming in on a specific IP address or for a specific host.

You cannot specify a wild card character for a server's IP address.

Note: If you want full path translation, including *query_string*, you must have an asterisk (*) in both the *request-template* and in the */file/path:function_name* as shown in the second example.

Example:

```
Service  /index.html /ics/api/bin/icsext05.so:serve_req

Service  /cgi-bin/hexcalc* /ics/api/calculator:HEXcalc*
```

Initial configuration file setting: None.

Program default setting: None.

PICSDBLookup - Customize the PICS label retrieval step

Use this directive to specify a customized application function you want the server to call to retrieve PICS labels for a specified URL. Your function can either dynamically create a PICS label for the requested document or search for a PICS label in an alternative file or database.

The format of the directive is:

```
PICSDBLookup
/path/file:function_name
```

/path/file
The fully qualified file name of your compiled program, including the extension.

function_name
The names you gave your application functions within your program.

Example:

```
PICSDBLookup  /api/bin/icsext05.so:get_pics
```

Initial configuration file setting: None.

DataFilter - Customize the Data Filter step

Use this directive to specify a customized application function you want the server to call during the Data Filter step. This code would provide three application functions:

- An *open* function to perform any initialization prior to processing the data
- A *write* function to process the data
- A *close* function to perform any clean up activities

You can only have one DataFilter active for each instance of the server.

The format of the directive is:

```
DataFilter /path/file:function_name:function_name:function_name
```

/path/file
The fully qualified file name of your compiled program, including the extension.

function_names
The names you gave your application functions within your program. You will need to supply the name of the open, write, and close functions.

Example:

```
DataFilter /ics/bin/icsext05.so:open_data:write_data:close_data
```

Initial configuration file setting: None.

Program default setting: None.

Log - Customize the Log step

Use this directive to specify a customized application function you want the server to call during the Log step. This code would supply logging and other processing you want performed after the connection has been closed.

The format of the directive is:

```
Log       request-template /path/file:function_name
```

request-template
: A template for requests that further determine if your application function is called. The specification can include the protocol, domain and host, can be preceded by a slash (/), and can use an asterisk (*) as a wildcard. For example, `/front_page.html`, `http://www.ics.raleigh.ibm.com`, /pub*, /*, and * are all valid.

/path/file
: The fully qualified file name of your compiled program, including the extension.

function_name
: The name you gave your application function within your program. You must supply the names of the open, write, and close functions.

Example:

```
Log       /index.html /api/bin/icsextpgm.so:log_url
```

Initial configuration file setting: None.

Program default setting: None.

Error - Customize the Error step

Use this directive to specify a customized application function you want the server to call during the Error step. This code would execute only when an error is encountered, to provide customized error routines.

The format of the directive is:

```
Error   request-template /path/file:function_name
```

request-template
: A template for requests that further determine if your application function is called. The specification can include the protocol, domain and host, can be preceded by a slash (/), and can use an asterisk (*) as a wildcard. For example, `/front_page.html`, `http://www.ics.raleigh.ibm.com`, /pub*, /*, and * are all valid.

/path/file
: The fully qualified file name of your compiled program, including the extension.

function_name
: The name you gave your application function within your program.

Example:

```
Error    /index.html /ics/api/bin/icsext05.so:error_rtns
```

Initial configuration file setting: None.

Program default setting: None.

PostExit - Customize the post-exit step

Use this directive to specify a customized application function you want the server to call during the post-exit step. This code will be executed regardless of the return codes from previous steps or other PostExit handlers. It allows you to clean up any resources allocated to process the request.

The format of the directive is:

```
PostExit    /path/file:function_name
```

/path/file
The fully qualified file name of your compiled program, including the extension.

function_name
The name you gave your application function within your program.

Examples::

```
PostExit     /ics/api/bin/icsext05.so:post_exit
```

Initial configuration file setting: None.

Program default setting: None.

ServerTerm - Customize the server termination step

Use this directive to specify a customized application function you want the server to call during the Server Termination step. This code would execute when an orderly shutdown occurs and whenever the server is restarted. It allows you to release resources allocated by a PreExit application function.

The format of the directive is:

```
ServerTerm   /path/file:function_name
```

/path/file
The fully qualified file name of your compiled program, including the extension.

function_name
The name you gave your application function within your program.

Example:

```
ServerTerm  /ics/api/bin/icsext05.so:shut_down
```

Initial configuration file setting: None.

Program default setting: None.

CounterDirectory - Specify a directory for the HTCounter program

Use this directive to specify the working directory for the HTCounter program. If this directive is not used, the default directory is *server_root*/Counters. For information on the HTCounter program, see "Using the HTCounter program to display page count, date, time, and text on a Web page" on page 209.

Examples

An example of an absolute path is:

```
CounterDirectory /mydirectory
```

An example of a relative path is:

```
CounterDirectory mydirectory
```

The relative path is appended to the value specified on the ServerRoot directive.

Initial configuration file setting

None.

Program default setting

CounterDirectory *server_root*/Counters

DebugToolAddr - Identify the workstation running the Remote Debugger

Use this directive to identify the workstation running the z/OS Debug Tool graphical user inteface (Remote Debugger). The Debug Tool is a Language Environment (LE) tool that can be used to debug your C/C++ GWAPI programs.

For information on using the Debug Tool with the Web server, see "Debugging C/C++ GWAPI programs" on page 375.

The format of the directive is:

```
DebugToolAddr IP_address or hostname [port_number]
```

For *IP_address or hostname*, enter the IP address or hostname of the workstation that is running the Remote Debugger. The default port number is 8000; specify a *port_number* if you are using a port other than the default.

Example

```
DebugToolAddr  127.20.5.3  8040
```

Initial configuration file setting

None.

Lightweight Directory Access Protocol - Set up shared configuration for the server

LDAPInfo - Define an external LDAP server

Use the LDAPInfo directive to provide the server with information about the LDAP servers in which to store information. Storing information on an external LDAP server allows applications to share the same information.

Note: In the configuration file, you must place LDAPInfo directives before any LDAPInclude directives or protection setups.

The format of the directive is:

```
LDAPInfo label-name  {
      subdirective value
      subdirective value
      subdirective value
```

```
      .
      .
      .
}
```

label-name
The name you want to associate with this LDAP server setup. The name can then be used by subsequent LDAPInclude and Protection directives to point to this LDAP setup.

subdirective
Put an LDAPInfo subdirective and its value on each line between the left and right brace. You cannot put any comment lines between the braces.

The LDAPInfo subdirectives are described below.

Example

```
LDAPInfo PrimaryLdapServer {
   Host ldap.ibm.com
   Transport TCP
   ClientAuthType Basic
   ServerAuthType Basic
   ServerDN "cn=HTTP Server, o=IBM c=US"
   ServerPasswordStashFile "StashFileName"
   UserSearchBase "o=IBM c=US"
   GroupSearchBase "o=IBM c=US"
}
```

Initial configuration file setting

None.

Program default setting

None.

LDAPInfo Subdirectives

Following are descriptions of the LDAPInfo subdirectives that can be used in an LDAP server setup. Four groups comprise the `LDAPInfo` sub-directives: general, timeout, server connection, and client connection.

General subdirectives

Use the following subdirectives to provide general information about the LDAP servers:

Host - Specify LDAP server hostname: Specify the hostname of the LDAP server.

Example:

```
Host ldap.mycompany.com
```

Transport - Specify LDAP connection protocol: Protocol to use to connect to the LDAP server. Possibilities are TCP(default) and SSL.

Example:

```
Transport TCP
```

Port - Specify LDAP server port number: Specify the port number the LDAP server listens on. The default port is 389 for TCP-transported connections, and 636 for SSL-transported connections. See “Transport - Specify LDAP connection protocol” for more information.

Example:

```
Port 389
```

Timeout subdirectives

When attempting to reach an external server, there is always a possibility the connection will take a long time to complete. The following subdirectives provide the server with timeout settings for connections to the LDAP server:

IdleConnTimeout - Specify how long to keep an idle LDAP connection: optional - Time during which to leave an idle LDAP connection open. The default is 10 minutes

Example:

```
IdleConnTimeout 10 minutes
```

WaitToRetryConnTime - Specify how long to wait before retrying LDAP connection: Amount of time to wait between unsuccessful attempts to connect to a server

Example:

```
WaitToRetryConn 5 minutes
```

SearchTimeout - Specify maximum LDAP search time: Time limit to wait for an LDAP search request to complete. The default is 10 seconds

Example:

```
SearchTimeout 2 minutes
```

CacheTimeout - Specify LDAP expiry time: Time-out of a cached entry. To reduce query time, the Web server caches user ID's, scrambled passwords, and group definitions. The Web server refreshes these User ID's, passwords, and group definitions anytime you do a restart of the Web server, even if the CacheTimeout has not expired. Use this directive to specify how long the server will return the cached copy as opposed to generating a new query. The default is ten minutes.

Example:

```
CacheTimeout 5 minutes
```

Server connection subdirectives

The following subdirectives provide the HTTP Server with parameters for establishing LDAP connections to read access control information.

KeyFileName - Specify LDAP key database: Specify the key database to use if the transport is Secure Sockets Layer (SSL). If the Web server establishes SSL connections to LDAP servers and to clients, the key file on the `KeyFileName` subdirective must match the key file on the `KeyFile` directive. The default value is the file specified by the KeyFile directive in the `httpd.conf` configuration file.

Example:

```
KeyFileName /usr/lpp/internet_server/base/key.kdb
```

KeyfilePasswordStashfile — Specify the name of the stash file created to hold the password to the key file: Specify the name of the stash file. The stash file holds the password to the key file.

Example: Use the **htadm** command to create the stash file. For example, issue the following command:

```
htadm -stash sfile secret
```

The command encrypts the password `secret` and stores it in the file, `sfile`.

Note: You cannot use the IKEYMAN utility or the gskkyman utility to create the stash file.

KeyLabel - Specify label of the certificate used for SSL connections: Specify a label for the certificate that the HTTP Server uses to authenticate with the LDAP server. If the Web server establishes SSL connections to LDAP servers and to clients, the label must match the label of the default certificate in the `.kdb` file.

Example:

```
KeyLabel "My Server Certificate"
```

ServerAuthType - Specify server LDAP authentication type: Authentication type for the server's connection to the LDAP server. Values are:

1. **None** — the LDAP server will allow anonymous access.

   Note: This option is supported for V3 LDAP servers only.
2. **Basic** — the HTTP Server logs in to the LDAP server. It uses its distinguished name from the `ServerDN` subdirective, and the password provided in the password stash file.

   Note: This option is supported for V2 and V3 LDAP servers.

Example:

```
ServerAuthType Basic
```

ServerDN - Specify distinguished name of the Web server: Distinguished name of the HTTP Server. This name is used when accessing the LDAP server when `ServerAuthType` is set to Basic.

Example:

```
ServerDN MyLdapServer
```

ServerPasswordStashFile - Specify LDAP password stash file: The file containing the encrypted password to access the LDAP server. The password is only used if `ServerAuthType` is Basic. Create the stash file using the htadm command or using the Configuration and Administration forms. See "htadm command" on page 389 for more information.

Example:

```
ServerPasswordStashFile /usr/lpp/internet_base/server/passwordFile
```

Client connection subdirectives

The following subdirectives provide the HTTP Server with parameters for establishing an LDAP connection on behalf of the client.

ClientAuthType - Specify client LDAP authentication type: Specify the authentication type for connections made on behalf of the client. Values are:

1. **Basic** - the client must provide a userid and password in order to authenticate
2. **Cert** - the client's certificate is used to authenticate.

   Note: In order for the client to successfully authenticate, the client must have used the https protocol, `SSLClientAuth` must be turned on, SSL client

authentication by the HTTP Server must be successful, and a search for the client's entry (using `UserCertFilter`) must be successful.

3. **BasicIfNoCert** -client's certificate is used for authentication. However, if https is not used, `SSLClientAuth` is not on, or if SSL client authentication by the HTTP Server fails, then basic authentication is used.

Example:

```
ClientAuthType BasicIfNoCert
```

UserSearchBase - Specify LDAP user search root: Specify the starting point for the LDAP server to search for user names.

Example:

```
UserSearchBase "o=IBM, c=US"
```

UserNameFilter - Specify LDAP user search filter: Specify the filter used to convert the username as input by the user to a search filter for an LDAP entry. The default is `"(&(objectclass=person)(cn=%v1* %v2*))"` where %v1 and %v2 are the words typed by the user.

For example, if the user types `"Pa Kel"`, the resulting search filter would be `"(cn=Pa* Kel*)"`. Search filters are described in "LDAP search filters" on page 297.

However, if multiple matching entries are returned, the HTTP Server does not know which to use, and fails authentication. If there were entries `(cn=Paul Kelsey)` AND `(cn=Paula Kelly)`, the above search string will return both entries and fail to authenticate.

Example:

```
UserNameFilter "(cn=%v1* %v2*)"
```

Note: The '%v#' syntax is only valid with UserCertFilter and UserNameFilter.

Use the **htadm** command to create the stash file. For example, issue the following command:

```
htadm -stash sfile secret
```

The command encrypts the password `secret` and stores it in the file, `sfile`.

Note: You cannot use the IKEYMAN utility or the gskkyman utility to create the stash file.

UserNameFieldSep - Specify delimiters for user data: Specify the set of characters used to separate the user's input into fields. The default value and the example below delimits data using a space, a comma, and the tab character

Example:

```
UserNameFieldSep " \t,"
```

UserCertFilter - Specify LDAP certificate search filter: Specify the certificate filter which converts the information in the client certificate passed over SSL to a search filter for an LDAP entry. The default is `"(&(objectclass=person)(cn=%v1)(ou=%v2)(o=%v3)(c=%v4))"`. SSL certificates include the following fields, all of which can be converted to a search filter:

1. common name

2. organizational unit
3. organization
4. country
5. locality
6. state or province
7. serial number

Example:

```
UserCertFilter "(cn=%v1)"
```

Note: The '%v#' syntax is only valid with UserCertFilter and UserNameFilter.

Use the **htadm** command to create the stash file. For example, issue the following command:

```
htadm -stash sfile secret
```

The command encrypts the password `secret` and stores it in the file, `sfile`.

Note: You cannot use the IKEYMAN utility or the gskkyman utility to create the stash file.

GroupSearchBase - Specify LDAP group search root: Specify the starting point for the LDAP server to search for group entries.

Example:

```
GroupSearchBase "o=IBM, c=US"
```

GroupNameFilter - Specify LDAP group search filter: Specify the filter LDAP uses to search for group names.

Example:

```
GroupNameFilter "(&(objectclass=groupNames)(objectclass=groupOfUniqueNames))"
```

Note: Because there is neither user input nor certificate input for this filter, the '%v#' syntax has no meaning. Do not use the '%v#' syntax with this filter.

GroupMemberAttrs - Specify attributes returned for group members: Specify the attributes of the group which contain member information. Many attributes may be specified separated by a comma. The default value and the example below show that member information is available in both the member and the uniqueMember attributes

Example:

```
GroupMemberAttrs "member,uniqueMember"
```

LDAPInclude - Retrieve configuration file information from the LDAP server

Use the `LDAPInclude` directive to provide the server with locations of configuration file information stored on the LDAP server.

The LDAPInfo directive must precede the LDAPInclude directive.

Format

`LDAPInclude` *`label filter attribute`*

label
name of the LDAP server setup defined in the LDAPInfo directive. The label tells the HTTP Server which LDAP server to use to locate configuration information related to the *attributes*

filter
LDAP search filter described in "LDAP search filters" on page 297, and RFC 1960

attribute
the name of the attribute whose value is some arbitrary part of the configuration file.

Example

```
LDAPInclude PrimaryLdapServer "(cn=web config)" description
```

The example will retrieve the value of the `description` attribute from a single entry with the common name `web config` on the server associated with `PrimaryLdapServer`.

Initial configuration file setting

None.

Program default setting

None.

Logging and Reporting - Customize logs and generate reports

By default, the Web server automatically generates reports using the HTLOGREP reporting program, unless you specify a third-party reporting program. To specify a third-party program, use the LoggingReportingProgram directive. To use logging and reporting functions, the DoReporting directive must be set on.

DoReporting — Use logging and reporting options

Use this directive to turn Web server logging and reporting options on or off:

- If DoReporting is set on, the Web server automatically generates reports using the default reporting program, HTLOGREP, unless you specify a third-party reporting program. To specify a third-party program, use the LoggingReportingProgram directive.

  Note: DoReporting must be set on for Access and Error logs to be purged.
- If DoReporting is set off, the Web server does not generate reports and ignores settings on the following Logging and Reporting directives:

  ```
  AccessLogArchive
  AccessLogExpire
  AccessLogSizeLimit,
  AccessReportDescription
  AccessReportDoDnsLookup
  AccessReportExcludeHostName
  AccessReportExcludeMethod
  AccessReportExcludeReturnCode
  AccessReportExcludeURL
  AccessReportRoot
  AccessReportTemplate
  AccessReportTopList
  ErrorLogArchive
  ErrorLogExpire
  ErrorLogSizeLimit
  LoggingReportingDebugOutput
  ```

Logging and reporting directives

```
LoggingReportingProgram
LoggingReportingProgramOptions
ReportDataArchive
ReportDataExpire
ReportDataSizeLimit
ReportProcessOldLogs
ReportDataCompressionProgram
ReportDataCompressionSuffix
ReportDataUnCompressionProgram
```

For an overview of reporting options, see Chapter 11, "Customizing logs and reports" on page 189.

Example

```
DoReporting Off
```

Initial configuration file setting

```
DoReporting On
```

Program default setting

```
DoReporting On
```

AccessLog - Name the path for the access log file

Use this directive to specify the place where the server logs all requests made by the client.

The server starts a new log file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on a given day. When creating the file, the server uses the file name you specify and appends a date suffix. The date suffix is in the format *Mmmddyyyy*, where *Mmm* is the first three letters of the month; *dd* is the day of the month; and *yyyy* is the year.

It is a good idea to remove old log files, because they can take up a significant amount of space. For information about removing old log files, refer to "AccessLogArchive - Remove existing access, agent, or referer log files or run a user exit" on page 493.

Example

```
AccessLog  logs/accesslog
```

Initial configuration file setting

```
AccessLog /usr/lpp/internet/server_root/logs/httpd-log
```

The example gives you *ServerRoot*/logs/httpd-log.*datesuffix.file_extension*. In other words, if *ServerRoot* is /usr/lpp/internet/server_root/ and the server is started on January 15, 1996 the result is:

```
/usr/lpp/internet/server_root/logs/httpd-log.Jan151996.extension
```

where *extension* is the web server generated file extension.

Program default setting

None.

AccessLogArchive - Remove existing access, agent, or referer log files or run a user exit

Values specified on the AccessLogArchive directive apply to access, agent, and referer logs. The collective size includes the size of all access logs or all agent logs or all referer logs, not the collective size of all types of logs.

At midnight each night, the server closes the current log and creates a new log file for the coming day. You can choose to do one of the following actions with the closed logs:

- Remove log files of a given age or when a given amount of storage is used by the collection of log files
- Allow closed logs to remain on your file system
- Branch to a user exit

To remove access, agent, or referer logs of a given age, specify this directive, in addition to the AccessLogExpire directive. To remove logs when their collective size exceeds a certain amount of storage, specify this directive, in addition to the AccessLogSizeLimit directive.

To allow closed logs to remain on your file system, you can accept the default, which is `AccessLogArchive none`.

To branch to a user exit, specify the path to the user exit and any parameters for the user exit on the AccessLogArchive directive. The server will append to this directive the path to the access, agent, or referer log.

The AccessLogArchive directive can be specified in any of the following formats:

```
AccessLogArchive   purge
AccessLogArchive   none
AccessLogArchive   userexit path_to_the_user-exit_program
                   [parameters for the user-exit]
```

Note: The directive must be typed on one line, even though it is shown here on two lines.

purge
Remove access log files of a given age or when their collective size exceeds a given amount of storage.

none
Do not remove access log files. `none` is the default.

userexit
Specifies the path of the user-exit program you want to branch to. You can optionally specify the parameters for your user-exit program, as shown in the following examples. The server appends the path to the access log to the directive.

Examples:

```
AccessLogArchive purge
AccessLogArchive none
AccessLogArchive userexit /u/userxyz/bin/backup/backup.rexx -d -a
```

For the `AccessLogArchive userexit` example, the user exit invocation is:

```
AccessLogArchive userexit /u/userxyz/bin/backup/backup.rexx
  -d -a /www/logs/httpd-log
```

Note: In this example the AccessLogArchive directive appears on two lines for printing purposes. The directive appears on one line in the `httpd.conf` file.

Initial configuration file setting

```
AccessLogArchive none
```

Program default setting

```
AccessLogArchive none
```

AccessLogExcludeURL - Suppress log entries for specific files or directories

Use this directive to specify that you do not want to log access requests made for specific files or directories that match a given URI template. For example, you might not want to log access requests for GIF files or you might not want to log access requests to a particular file or directory on your server.

You can have multiple occurrences of this directive in your configuration file. You can also put multiple entries for the same directive if you separate them by one or more spaces.

Example

```
AccessLogExcludeURL  *.gif

AccessLogExcludeURL  /Freebies/*
```

Initial configuration file setting

None. The server includes in the access log requests for all files and directories.

Program default setting

None.

AccessLogExcludeMethod - Suppress log entries for files or directories requested by a given method

Use this directive to specify that you do not want to log access requests made for files or directories by using a specific method. For example, you might not want to log DELETE requests for files or directories.

You can have multiple occurrences of this directive in your configuration file. You can also put multiple methods on the same directive if you separate them by one or more spaces.

Example

```
AccessLogExcludeMethod GET
AccessLogExcludeMethod PUT
AccessLogExcludeMethod POST
AccessLogExcludeMethod DELETE
```

Initial configuration file setting

None. The server includes in the access log the files and directories requested by all types of methods.

Program default setting

None.

AccessLogExcludeMimeType - Suppress log entries for specific MIME types

Use this directive to specify that you do not want to log access requests made for directories or files of a given MIME type. (Examples of MIME types are text/html, image/gif, and image/jpeg.) For example, you might not want to log access requests for GIF images.

You can have multiple occurrences of this directive in your configuration file. You can also put multiple MIME types on the same directive if you separate them by one or more spaces.

Example

```
AccessLogExcludeMimeType image/gif
```

Initial configuration file setting

None. The access log includes requests to the server for files and directories of all MIME types.

Program default setting

None.

AccessLogExcludeReturnCode - Suppress log entries for specific return codes

Use this directive to specify that you do not want to log access requests that fall within a given range of error code numbers. These error code numbers are HTTP status codes. You cannot specify individual codes. For example, specifying 300 indicates that you want to exclude all access requests that fall within the range 300-399. To exclude these requests, you would specify:

```
AccessLogExcludeReturnCode 300
```

You can have multiple occurrences of this directive in your configuration file. You can also put multiple return codes on the same directive if you separate them by one or more spaces.

Example

```
AccessLogExcludeReturnCode 300
```

Initial configuration file setting

None. The access log includes all requests to the server, regardless of the code.

Program default setting

None.

AccessLogExpire - Remove existing access log files when they reach a given age in days

Use this directive to specify that you want to remove access log files when they reach a certain age (in days).

This directive requires that you also specify the AccessLogArchive directive, described under “AccessLogArchive - Remove existing access, agent, or referer log files or run a user exit” on page 493. You can have only one occurrence of this directive in your configuration file.

The format of the AccessLogExpire directive is:

```
AccessLogExpire   number-of-days
```

number-of-days
Specifies that access logs older than this value are to be removed. *number-of-days* must be an integer; decimal values such as 1.5 are not valid. The default is 0, a value that indicates that no expiration date exists.

The file creation date, as reported by the operating system, is used to determine the age of the access log file. The suffix of the filename, such as `httpd-log.Mar221996.extension`, is not used to determine file age. (*extension* is the file extension.)

Example

```
AccessLogExpire 10
```

Initial configuration file setting

```
AccessLogExpire 0
```

Program default setting

```
AccessLogExpire 0
```

AccessLogSizeLimit - Remove existing access log files when they reach a given collective size

Use this directive to specify that you want to remove access log files when they reach a collective size (in megabytes).

This directive requires that you also specify the AccessLogArchive directive, described under "AccessLogArchive - Remove existing access, agent, or referer log files or run a user exit" on page 493. You can have only one occurrence of this directive in your configuration file.

The format of the AccessLogSizeLimit directive is:

```
AccessLogSizeLimit   number-of-megabytes
```

number-of-megabytes
Specifies that when the combined size of the access log files exceeds this value, files are deleted starting with the oldest file, until the collective size is within the limit specified on the AccessLogSizeLimit directive. *number-of-megabytes* must be an integer. The default is 0, a value that indicates that no access log files are to be removed.

This directive takes effect after the AccessLogExpire directive has taken effect.

Example

```
AccessLogSizeLimit 4
```

Initial configuration file setting

```
AccessLogSizeLimit 0
```

Program default setting

```
AccessLogSizeLimit 0
```

AccessReportDescription - Give a short description of the HTLOGREP report to be created

Use this directive to include a short description of the report to be created with this template.

Note: Use this directive only if you are using the HTLOGREP program to produce your reports.

Example

```
AccessReportDescription   Report on Web page accesses
```

Initial configuration file setting

None.

Program default setting

None.

AccessReportDoDnsLookup- Display client hostnames in HTLOGREP access reports

Use this directive to display client hostnames in access reports created by HTLOGREP.

Note: Use this directive only if you are using the HTLOGREP program to produce your reports.

When access reports are created, a list of the client hostnames is generated and DNS is called. This results in improved performance since the server does not need to do a DNS lookup for each client access.

Example

```
AccessReportDoDnsLookup   On
```

Initial configuration file setting

```
AccessReportDoDnsLookup  Off
```

Program default setting

```
AccessReportDoDnsLookup  Off
```

AccessReportExcludeURL - Suppress log entries for specific files or directories from the HTLOGREP report

Use this directive to specify that you do not want the HTLOGREP program to include specific files or directories that match a given URI template. For example, you might not want to include requests for GIF files or access requests to a particular file or directory on your server.

Note: Use this directive only if you are using the HTLOGREP program to produce your reports.

Examples

```
AccessReportExcludeURL *.gif
AccessReportExcludeURL oldfiles*
```

You can only specify one file or directory on a directive. However, you can have multiple occurrences of this directive in your configuration file.

The following rules are used when attempting to match the template against a URL in the report being generated:

- If the template has no slash, any directories in the URL are ignored, and only the filename part of the URL is used for matching.

- When matching filenames, any number of asterisks can be used as wildcards, which can span dots (.), but not slashes (/).
- If the template contains any slash, it is considered to be path-specific. Therefore, the number of directories in the URL must match the number of directories in the template, and each directory name must match. When matching directory names, any number of asterisks can be used as wildcards, which can span dots (.).

Initial configuration file setting
None.

Program default setting
None.

AccessReportIncludeURL - Include only log entries for specific files or directories in the HTLOGREP report

Use this directive to specify that you want the HTLOGREP program to include only access requests made for specific files or directories that match a given URI template. For example, you might want to include only access requests for HTML files or you might want to include access requests to a particular file or directory on your server.

Note: Use this directive only if you are using the HTLOGREP program to produce your reports.

Example

```
AccessReportIncludeURL /*.html
```

You can only specify one file name or directory on a directive. However, you can have multiple occurrences of this directive in your configuration file.

The following rules are used when attempting to match the template against a URL in the report being generated:

- If the template has no slash, any directories in the URL are ignored, and only the filename part of the URL is used for matching.
- When matching filenames, any number of asterisks can be used as wildcards, which can span dots (.), but no slashes (/).
- If the template contains any slash, it is considered to be path-specific. Therefore, the number of directories in the URL must match the number of directories in the template, and each directory name must match. When matching directory names, any number of asterisks can be used as wildcards, which can span dots (.).

Initial configuration file setting
None.

Program default setting
None.

AccessReportExcludeHostName - Suppress the log entries for specific host names from the HTLOGREP report

Use this directive to specify that the HTLOGREP program should not include in the access report requests made by host names or IP addresses that match a given template.

Note: Use this directive only if you are using the HTLOGREP program to produce your reports.

You can only specify one host name or IP address on a directive. However, you can have multiple occurrences of this directive in your configuration file.

Note: To exclude host names, you must set the DNS-Lookup directive to `On`. If the DNS-Lookup directive is set to `Off` (the default), you can exclude IP addresses only.

Example

```
AccessReportExcludeHostName 9.85.*.*
AccessReportExcludeHostName *.edu
```

Initial configuration file setting

None.

Program default setting

None.

AccessReportIncludeHostName - Include only log entries for specific host names in the HTLOGREP report

Use this directive to specify that the HTLOGREP program should include in the access report requests made by host names or IP addresses that match a given template.

Note: Use this directive only if you are using the HTLOGREP program to produce your reports.

You can only specify one host name or IP address on a directive. However, you can have multiple occurrences of this directive in your configuration file.

Note: To include host names, you must set the DNS-Lookup directive to `On`. If the DNS-Lookup directive is set to `Off` (the default), you can include IP addresses only.

Example

```
AccessReportIncludeHostName 9.9.99.*
AccessReportIncludeHostName *.com
```

Initial configuration file setting

None.

Program default setting

None.

AccessReportExcludeMethod - Suppress the log entries of a given method type from the HTLOGREP report

Use this directive to specify that the HTLOGREP program should not include in the access report requests of a given method type.

Note: Use this directive only if you are using the HTLOGREP program to produce your reports.

You can specify only one method type on a directive. However, you can have multiple occurrences of this directive in your configuration file.

Example

```
AccessReportExcludeMethod GET
AccessReportExcludeMethod PUT
AccessReportExcludeMethod POST
```

Initial configuration file setting

None.

Program default setting

None.

AccessReportExcludeReturnCode - Suppress the log entries with a given return code from the HTLOGREP report

Use this directive to specify that the HTLOGREP program should not include in the access report requests that fall within a given set of error code numbers.

Notes:

1. The error code numbers are http status codes. You cannot specify individual return codes. Specifying 300 indicates that you want to exclude from the report access requests with redirection return codes (301, 302, 303, and 304).
2. You can only specify one set of return codes on a directive. However, you can have multiple occurrences of this directive in your configuration file.
3. Use this directive only if you are using the HTLOGREP program to produce your reports.

Example

```
AccessReportExcludeReturnCode 200
AccessReportExcludeReturnCode 400
```

Initial configuration file setting

None.

Program default setting

None.

AccessReportRoot - Name the path for the root directory where HTLOGREP access log reports are stored

Use this directive to specify the path and file name where you want the server to store HTLOGREP access log reports and summary databases.

Note: Use this directive only if you are using the HTLOGREP program to produce your reports.

We recommend that you accept the default path. If you choose to specify a different path, you will need to create the new directory with all the appropriate permissions and add a PASS directive to enable the server to honor requests for reports in that directory.

If you are running with workload management enabled, you should have unique AccessReportRoot directives for each instance of the server based on the subsystem name. If you specify -SN system1, you should have AccessReportRoot /usr/lpp/internet/server_root/pub/reports/system1 and a corresponding Pass directive.

Example

```
AccessReportRoot WWW/reports
```

Initial configuration file setting

```
AccessReportRoot /copy/usr/lpp/internet/server_root/pub/reports
```

Program default setting

None.

AccessReportTemplate - Name the HTLOGREP report template

Use this directive to specify the name of the HTLOGREP report template. The default template is named "Top50".

Note: Use this directive only if you are using the HTLOGREP program to produce your reports.

The format of the AccessReportTemplate is:

```
AccessReportTemplate report_title
```

report_title
The name of the report. The name cannot include any blanks.

Example

```
AccessReportTemplate Page_Hits
```

Initial configuration file setting

```
AccessReportTemplate  Top50    {
AccessReportDescription        Top 50 files and visitors
AccessReportTopList            50
                               }
```

Program default setting

None.

AccessReportTopList - Specify the top number of items on which HTLOGREP is to report

Use this directive to specify the top number of items on which the HTLOGREP program is to report.

Note: Use this directive only if you are using the HTLOGREP program to produce your reports.

The format of the AccessReportTopList is:

```
AccessReportTopList top_number|all
```

top_number
Specifies that the report is to include the *top_number* most frequently occurring entries in the access log. This must be an integer value.

Note: When you are using Web usage mining, the value for the AccessReportTopList directive must be an integer value up to 1000.

all Specifies that the report is to include all entries in the report.

Example

```
AccessReportTopList 10
```

Initial configuration file setting

None.

Program default setting

None.

AgentLog - Name the path for the agent log file

Use this directive to specify the place where the server logs statistics about which Web browser was used to access a Web page. By default the server writes an entry to this log each time a client sends the server a request. For every entry made in the access log, the agent log has a corresponding entry that indicates the browser used to display the page or file requested by the client.

The server starts a new agent log file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on a given day. When creating the file, the server uses the file name you specify and appends a date suffix. The date suffix is in the format *Mmmddyyyy*, where *Mmm* is the first three letters of the month; *dd* is the day of the month; and *yyyy* is the year.

Example

```
AgentLog  logs/agent-log
```

Initial configuration file setting

```
AgentLog /usr/lpp/internet/server_root/logs/agent-log
```

This gives you *ServerRoot*/logs/agent-log.*datesuffix.file_extension*. In other words, if *ServerRoot* is /usr/lpp/internet/server_root/ and the server is started on January 15, 1996 the result is:

```
/usr/lpp/internet/server_root/logs/agent-log.Jan151996.extension
```

where *extension* is the web server generated file extension.

Program default setting

None.

CacheAccessLog - Specify the path for the cache access log files

If the server is running as a proxy, you can log requests to the cache separately from other requests. Use the CacheAccessLog directive to specify the path and file name where you want the server to put access requests for cached files. To enable logging of requests to the proxy cache, the following directives must be defined:

- Caching must be turned ON (default is OFF)
- CacheRoot (by default, no CacheRoot is defined)
- CacheAccessLog

The value of CacheAccessLog can either be an absolute path or a path relative to ServerRoot (one example is shown of each).

Note: If you choose to use CacheAccessLog, access requests for cached files are logged, but they are not included in the access reports. Access reports contain only information from access logs, not from cache access logs. Therefore, if you want access reports to contain access requests for cached files, do *not* specify the CacheAccessLog directive.

The server starts a new log file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on a given day. When creating the file, the server uses the file name you specify and appends a date

suffix. The date suffix is in the format *Mmmddyyyy*, where *Mmm* is the first three letters of the month; *dd* is the day of the month; and *yyyy* is the year.

The format of this directive is

```
CacheAccessLog  <file_path>
```

Example

```
CacheAccessLog  /absolute/path/logfile
CacheAccessLog  logs/logfile
```

Initial configuration file setting

None. The server does not log cache access requests if you include this directive in your configuration file.

Program default setting

None.

CgiErrorLog - Name the path for the CGI error log file

Use this directive to specify the place where the server logs standard error output (stderr) from CGI programs.

The server starts a new CGI error file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on a given day. When creating the file, the server uses the file name you specify and appends a date suffix. The date suffix is in the format *Mmmddyyyy*, where *Mmm* is the first three letters of the month; *dd* is the day of the month; and *yyyy* is the year.

Example

```
CgiErrorLog logs/cgi-error
```

Initial configuration file setting

```
CgiErrorLog /usr/lpp/internet/server_root/logs/cgi-error
```

This gives you *ServerRoot*/logs/cgi-error.*datesuffix.file_extension*. In other words, if *ServerRoot* is /usr/lpp/internet/server_root/ and the server is started on January 15, 1996 the result is:

```
/usr/lpp/internet/server_root/logs/cgi-error.Jan151996.extension
```

where *extension* is the web server generated file extension.

Program default setting

None.

ErrorLog - Name the file where you want to log internal server errors

Use this directive to specify the path and file name where you want the server to log internal errors.

The server starts a new log file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on a given day. When creating the file, the server uses the file name you specify and appends a date suffix. The date suffix is in the format *Mmmddyyyy*, where *Mmm* is the first three letters of the month; *dd* is the day of the month; and *yyyy* is the year.

Example

```
ErrorLog logs/errorlog
```

Initial configuration file setting

```
ErrorLog /usr/lpp/internet/server_root/logs/httpd-error
```

Program default setting

None.

ErrorLogArchive - Remove existing error or CGI error log files or run a user exit

Values specified on the ErrorLogArchive directive apply to error and CGI error logs. The collective size includes the size of all error logs or all CGI logs, not the collective size of both types of logs.

At midnight each night, the server closes the current error and CGI error logs and creates new log files for the coming day. You can choose to do one of the following actions with the closed error logs:

- Remove log files of a given age or when a given amount of storage is used by the collection of error log files
- Allow closed logs to remain in your file system
- Branch to a user exit.

To remove logs of a given age, specify this directive, in addition to the ErrorLogExpire directive. To remove logs when their collective size exceeds a certain amount of storage, specify this directive, in addition to the ErrorLogSizeLimit directive.

To allow closed logs to remain on your file system, you can accept the default, which is `ErrorLogArchive none`.

To branch to a user exit, specify the path to the user exit and any parameters for the user exit on the ErrorLogArchive directive. The server will append to this directive the path to the error or CGI error log.

The ErrorLogArchive directive can be specified in any of the following formats:

```
ErrorLogArchive   purge
ErrorLogArchive   none
ErrorLogArchive   userexit path_to_the_user-exit program
                  [parameters for the user-exit]
```

Note: The directive must be typed on one line, even though it is shown here on two lines.

purge
Remove log files of a given age or when their collective size exceeds a given amount of storage.

none
Do not remove log files. `none` is the default.

userexit
Specifies the path of the user-exit program you want to branch to. You can optionally specify the parameters for your user-exit program, as shown in the following examples. The server appends the path to the error log to the directive.

Examples:

```
ErrorLogArchive purge
ErrorLogArchive none
ErrorLogArchive userexit /u/userxyz/bin/errback/backup.rexx -d -a
```

For the `ErrorLogArchive userexit` example, the user exit invocation is:

```
ErrorLogArchive userexit /u/userxyz/bin/errback/backup.rexx
-d -a /www/logs/httpd-error
```

Note: The directive must be typed on one line, even though it is shown here on two lines.

Initial configuration file setting

```
ErrorLogArchive none
```

Program default setting

```
ErrorLogArchive none
```

ErrorLogExpire - Remove existing error log files when they reach a given age in days

Use this directive to specify that you want to remove error log files when they reach a certain age (in days).

This directive requires that you also specify the ErrorLogArchive directive, described under "ErrorLogArchive - Remove existing error or CGI error log files or run a user exit" on page 504. You can have only one occurrence of this directive in your configuration file.

The format of the ErrorLogExpire directive is:

```
ErrorLogExpire   number-of-days
```

number-of-days
Specifies that error logs older than this value are to be removed. *number-of-days* must be an integer; decimal values such as 1.5 are not valid. The default is 0, a value that indicates that no expiration date exists.

The file creation date, as reported by the operating system, is used to determine the age of the error log file. The suffix of the filename, such as `httpd-log.Mar221996`, is not used to determine file age.

Example

```
ErrorLogExpire 10
```

Initial configuration file setting

```
ErrorLogExpire 0
```

Program default setting

```
ErrorLogExpire 0
```

ErrorLogSizeLimit - Remove existing error log files when they reach a given collective size

Use this directive to specify that you want to remove error log files when they reach a collective size (in megabytes).

This directive requires that you also specify the ErrorLogArchive directive, described under "ErrorLogArchive - Remove existing error or CGI error log files or run a user exit" on page 504. You can have only one occurrence of this directive in your configuration file.

The format of the ErrorLogSizeLimit directive is:

```
ErrorLogSizeLimit   number-of-megabytes
```

number-of-megabytes
Specifies that when the sum total size of the error log files exceeds this value, files are deleted starting with the oldest file, until the collective size is within the limit specified on the ErrorLogSizeLimit directive. *number-of-megabytes* must be an integer. The default is 0, a value that indicates that no error log files are to be removed.

This directive takes effect after the ErrorLogExpire directive has taken effect.

Example

```
ErrorLogSizeLimit 4
```

Initial configuration file setting

```
ErrorLogSizeLimit 0
```

Program default setting

```
ErrorLogSizeLimit 0
```

LogFormat - Specify common or old log file format

Use this directive to specify whether you want your server to write log files in the common format or old format.

If you plan to use the reporting functions described under "Tailoring the reports your server creates" on page 196, you must accept the default file format, common.

The common format is the one used by most Web servers. The old format was used with early versions of Web servers from CERN. You most likely will want to use the common format, which is the default.

Example

```
LogFormat Old
```

Initial configuration file setting

```
LogFormat Common
```

Program default setting

```
LogFormat Common
```

LoggingReportingDebugOutput— Generate debug log for HTLOGREP reporting program

Use this directive to help you troubleshoot problems with the HTLOGREP reporting program. If you specify a debug log file name using this directive, an HTLOGREP debug log will be created at midnight. To use this option, the DoReporting directive must be set on.

Example

```
LoggingReportingDebugOutput  /temp/mydebug
```

Initial configuration file setting

None

Example of HTLOGREP Debug Information Log

```
Configuration: report_root is /usr/lpp/internet/server_root/pub/reports
Configuration: access_log is /usr/lpp/internet/server_root/logs/httpd-log
Configuration: DoDnsLookup is FALSE

-----------------------------------------------------------------------------
Reporting: Name of Config Template: Top50
Reporting: Description:  Top 50 most frequently requested files and most
           frequent visitors
-----------------------------------------------------------------------------

Archiving: Entering ArchiveLog()
Archiving: Config File: /etc/httpd.conf
Archiving: Access Log Directive: /usr/lpp/internet/server_root/logs
   /httpd-log
Archiving: Error Log Directive: /usr/lpp/internet/server_root/logs
   /httpd-errors
Archiving: Agent Log Directive: /usr/lpp/internet/server_root/logs
   /agent-log
Archiving: Referer Log Directive: /usr/lpp/internet/server_root/logs
   /referer-log
Archiving: CgiError Log Directive: /usr/lpp/internet/server_root/logs
   /cgi-error
Archiving: AccessLogArchive purge
Archiving: ErrorLogArchive none
Archiving: AccessLogExpire 1
Archiving: ErrorLogExpire 0
Archiving: AccessLogSizeLimit 0
Archiving: ErrorLogSizeLimit 0
Archiving: ReportDataArchive none
Archiving: ReportDataExpire 0
Archiving: ReportDataSizeLimit 0
Archiving: Configuration read successfully
Archiving: Beginning processing of AccessLog
Archiving: Today Time = 940002649 day = 10879
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /httpd-log.Sep171999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /httpd-log.Sep171999,
28 days old
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /httpd-log.Sep231999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /httpd-log.Sep231999,
22 days old, 3 Kbytes
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /httpd-log.Sep241999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /httpd-log.Sep241999,
21 days old, 1 Kbytes
...Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /httpd-log.Sep251999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /httpd-log.Sep251999,
20 days old, 1 Kbytes
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
  /httpd-log.Sep261999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /httpd-log.Sep261999,
19 days old, 1 Kbytes
Archiving: Clean by date started, days = 1:
Archiving: Clean by date: Successfully removed /usr/lpp/internet
   /server_root/logs/
```

```
httpd-log.Sep171999
Archiving: Clean by date: Successfully removed /usr/lpp/internet
   /server_root/logs/
httpd-log.Sep231999
Archiving: Clean by date: Successfully removed /usr/lpp/internet
   /server_root/logs/
httpd-log.Sep241999
Archiving: Clean by date: Successfully removed /usr/lpp/internet
   /server_root/logs/
httpd-log.Sep251999
Archiving: Clean by date: Successfully removed /usr/lpp/internet
   /server_root/logs/
httpd-log.Sep261999
...
Archiving: Beginning processing of AgentLog
Archiving: Today Time = 940002649 day = 10879
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /agent-log.Sep171999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /agent-log.Sep171999,
28 days old
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /agent-log.Sep231999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /agent-log.Sep231999,
22 days old, 2 Kbytes
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /agent-log.Sep241999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /agent-log.Sep241999,
21 days old, 1 Kbytes

Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /agent-log.Sep251999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /agent-log.Sep251999,
20 days old, 1 Kbytes
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /agent-log.Sep261999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /agent-log.Sep261999,
19 days old, 1 Kbytes
Archiving: Clean by date started, days = 1:
Archiving: Clean by date: Successfully removed /usr/lpp/internet
   /server_root/logs/
agent-log.Sep171999
Archiving: Clean by date: Successfully removed /usr/lpp/internet
   /server_root/logs/
agent-log.Sep231999
Archiving: Clean by date: Successfully removed /usr/lpp/internet
   /server_root/logs/
agent-log.Sep241999
Archiving: Clean by date: Successfully removed /usr/lpp/internet
   /server_root/logs/
agent-log.Sep251999
Archiving: Clean by date: Successfully removed /usr/lpp/internet
   /server_root/logs/
agent-log.Sep261999
...
Archiving: Beginning processing of RefererLog
Archiving: Today Time = 940002649 day = 10879
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /referer-log.Sep171999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /referer-log.Sep171999,
28 days old
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
```

```
   /referer-log.Sep231999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /referer-log.Sep231999,
22 days old, 2 Kbytes
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /referer-log.Sep241999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /referer-log.Sep241999,
21 days old, 1 Kbytes
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /referer-log.Sep251999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /referer-log.Sep251999,
20 days old, 1 Kbytes
Archiving: stat() OK for file /usr/lpp/internet/server_root/logs
   /referer-log.Sep261999
Archiving: adding to File list: /usr/lpp/internet/server_root/logs
   /referer-log.Sep261999,
19 days old, 1 Kbytes
Archiving: Leaving ArchiveLog() - archiving complete
```

Note: Many of the statements in the example information log appear on two lines for printing purposes. Each of these statements appears on one line in your log.

LoggingReportingProgram — Specify the reporting program to be used

Use this directive if you want to use a third-party reporting program or change the default options for the HTLOGREP reporting program:

- To use a third-party reporting program, such as Analog, instead of HTLOGREP:
  1. Enter the complete file path of the third-party reporting program.
  2. Specify reporting program options using the LoggingReportingProgramOptions directive.
- If you are using the default reporting program, HTLOGREP, but want to change the default reporting options:
  1. Enter the complete file path of the HTLOGREP program. The default path is `/usr/lpp/internet/sbin/htlogrep`.
  2. Specify HTLOGREP options using the LoggingReportingProgramOptions directive.

For an overview of reporting options, see "Tailoring the reports your server creates" on page 196.

Example if using third-party program

```
LoggingReportingProgram  /usr/lpp/internet/sbin/analog
```

Example if using HTLOGREP and changing default options

```
LoggingReportingProgram  /usr/lpp/internet/sbin/htlogrep
```

Initial configuration file setting

```
None
```

Program default setting

```
None.
```

LoggingReportingProgramOptions — Specify reporting program options

Use this directive to:

- Specify options for a third-party reporting program you specified on the LoggingReportingProgram directive. See your program documentation for information on possible options.
- Change the default reporting options for the HTLOGREP reporting program. If you specify options for HTLOGREP using this directive, you must also specify the complete file path of the HTLOGREP program on the LoggingReportingProgram directive.

  Possible options are:
  - Use a configuration file other than the default configuration file (**-c**).
  - Create a debug log (**-d**). For an example, see "Example of HTLOGREP Debug Information Log" on page 507.
  - Specify a log file name other than the default (**-l**).
  - If you are running the Web server in Scalable Server mode, specify a subsystem name (**-s**).
  - If you are running the Web server in Scalable Server mode, specify a wait period in minutes (**-w**). The default wait period is 5 minutes. For no wait, specify **-w 0**.

For an overview of reporting options, see "Tailoring the reports your server creates" on page 196.

Example if using HTLOGREP and changing default options

```
LoggingReportingProgramOptions  "-c/etc/myhttpd.conf -d/temp/mydebug -l/temp/mylog"
```

Note: Any option or set of options that contains a blank must be enclosed in quotes.

Initial configuration file setting

```
None
```

Program default setting

```
None
```

LogTime - Specify GMT or local time stamps in log files

Use this directive to specify whether your logs should record entries using local time or Greenwich Mean Time (GMT).

Example

```
LogTime GMT
```

Initial configuration file setting

```
LogTime LocalTime
```

Program default setting

LogTime LocalTime

LogToSyslog - Log access information to the syslog, in addition to or instead of the error log

Use this directive to specify whether you want your server to log access requests and errors to the MVS syslog daemon in addition to the access and error log files. If you want to have access and error log information sent to the syslog file in addition to sending it to the log files, you must change the default.

The MVS syslog daemon must be up and running on your system before you specify that error log information be written to it. You can choose whether to log only access or error information or to log both.

For information about how to configure the syslog daemon, refer to *TCP/IP for z/OS UNIX System Services MVS Application Feature Guide*, SC31-8069.

Example

```
LogToSyslog On
```

Initial configuration file setting

```
LogToSyslog Off
```

Program default setting

```
LogToSyslog Off
```

NoLog - Suppress log entries for specific hosts or domains matching a template

Use this directive to specify that you do not want to log access requests made from specific hosts or domains that match a given template. For example, you may not want to log access requests from local hosts.

You can have multiple occurrences of this directive in your configuration file. You can also put multiple templates on the same directive if you separate them by one or more spaces. You can use host names or IP addresses on the templates.

Note: To use host name templates, you must set the DNS-Lookup directive to `On`. If the DNS-Lookup directive is set to `Off` (the default), you can use IP address templates only.

Example

```
NoLog 128.141.*  *.edu  localhost.*
```

Initial configuration file setting

None.

Program default setting

None.

ProxyAccessLog - Name the path for the proxy access log file

Use this directive to specify the place where the server logs all requests which it handles through its proxy services.

The server starts a new log file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on a given day. When creating the file, the server uses the file name you specify and appends a date suffix. The date suffix is in the format *Mmmddyyyy*, where *Mmm* is the first three letters of the month; *dd* is the day of the month; and *yyyy* is the year.

It is a good idea to remove old log files, because they can take up a significant amount of space on your file system.

Example

```
ProxyAccessLog  logs/proxylog
```

Initial configuration file setting

```
ProxyAccessLog /usr/lpp/internet/server_root/logs/proxy-log
```

This gives you *ServerRoot*/logs/proxy-log *datesuffix.file_extension*. In other words, if *ServerRoot* is /usr/lpp/internet/server_root/ and the server is started on January 15, 1996 the result is:

```
/usr/lpp/internet/server_root/logs/proxy-log.Jan151996 extension
```

where *extension* is the web server generated file extension.

Program default setting

None.

RefererLog - Name the path for the referer log file

Use this directive to specify the place where you want the server log the identity of the Web page that referred to (linked to) the requested Web page. By default the server writes an entry to this log each time a client sends the server a request. For every entry made in the access log, the referer log has a corresponding entry that indicates which page referred to the page that was requested by the client. If no page referred to the requested page, the entry is two quotation marks (" ").

The server starts a new referer log file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on a given day. When creating the file, the server uses the file name you specify and appends a date suffix. The date suffix is in the format *Mmmddyyyy*, where *Mmm* is the first three letters of the month; *dd* is the day of the month; and *yyyy* is the year.

Example

```
RefererLog logs/referer-log
```

Initial configuration file setting

```
RefererLog /usr/lpp/internet/server_root/logs/referer-log
```

This gives you *ServerRoot*/logs/referer-log.*datesuffix.file_extension*. In other words, if *ServerRoot* is /usr/lpp/internet/server_root/ and the server is started on January 15, 1996 the result is:

```
/usr/lpp/internet/server_root/logs/referer-log.Jan151996.extension
```

where *extension* is the web server generated file extension.

Program default setting

None.

ReportDataCompressionProgram - Specify path to the compression program

Use this directive to specify the path to the compression program (such as PKZIP2, GZIP, or compress) and any program parameters for the compression program. Include any command line parameters on the same line. This compression program is to be used to compress access data log files.

Example

```
ReportDataCompressionProgram /bin/compress
```

Initial configuration file setting

None

Program default setting
None

ReportDataUnCompressionProgram - Specify path to the uncompression program

Use this directive to specify the path to the uncompression program (such as UNZIP, GZIP, or uncompress) and any program parameters for the uncompression program. Include any command line parameters on the same line. This uncompression program is to be used to uncompress access data log files.

Example
```
ReportDataUnCompressionProgram /bin/uncompress
```

Initial configuration file setting
None.

Program default setting
None.

ReportDataCompressionSuffix - Specify the suffix appended to the compressed report data files

Use this directive to specify the suffix appended to the compressed report data files.

Example
```
ReportDataCompressionSuffix .Z
```

Initial configuration file setting
None.

Program default setting
None.

ReportProcessOldLogs - Check for old logs in the log directory

Use this directive to indicate that you want the server to check for old access logs in the log directory that are not listed in the list of log files that have been processed into reports. With this directive, you can process old access log files by:
- Appending the data from the old access log files to existing reports
- Creating reports for all access log files and overwriting existing reports
- Creating a report for the most recently created access log file.

The format of the ReportProcessOldLogs directive is:
```
ReportProcessOldLogs append|force|last
```

append
Add to existing access log reports data from log files that were not originally included in the reports.

force
Overwrite existing access log reports with reports based on data from all access log files, regardless of whether they were originially included in the reports.

Note: The only way to erase reports named access.*mmddyyyy* files is to archive them with the ReportDataArchive directive, described under "ReportDataArchive - Specify whether to remove existing accessdata files".

last
Create reports based on data from the most recently created access log file.

Examples

```
ReportProcessOldLogs append
ReportProcessOldLogs force
ReportProcessOldLogs last
```

Initial configuration file setting

ReportProcessOldLogs append

Program default setting

ReportProcessOldLogs append

ReportDataSizeLimit - Remove existing access data files when they reach a given collective size

Use this directive to specify that you want to remove access data files when they reach a collective size (in megabytes).

This directive requires that you also specify the ReportDataArchive directive, described under "AccessLogArchive - Remove existing access, agent, or referer log files or run a user exit" on page 493. You can have only one occurrence of this directive in your configuration file.

The format of the ReportDataSizeLimit directive is:

```
ReportDataSizeLimit   number-of-megabytes
```

number-of-megabytes
Specifies that when the sum total size of the access data files exceeds this value, files are deleted starting with the oldest file, until the collective size is within the limit specified on the ReportDataSizeLimit directive. *number-of-megabytes* must be an integer. The default is 0, a value that indicates that no access data files are to be removed.

This directive takes effect after the ReportDataExpire directive has taken effect.

Example

```
ReportDataSizeLimit 4
```

Initial configuration file setting

```
ReportDataSizeLimit 0
```

Program default setting

ReportDataSizeLimit 0

ReportDataArchive - Specify whether to remove existing accessdata files

Use this directive to specify whether you want to remove existing access data log files.

If you want to remove access data files, you also need to specify the ReportDataExpire directive, described under "ReportDataExpire - Remove existing

access data files when they reach a given age in days". You can have only one occurrence of this directive in your configuration file.

Even after you remove access data files, the data from these files is still available for reports to use, until you specify the ReportProcessOldLogs directive with the `force` option.

The ReportDataArchive directive can be specified in any of the following formats:

```
ReportDataArchive purge
ReportDataArchive none
ReportDataArchive userexit path_to_the_user-exit_program
                           [parameters for the user-exit]
```

Note: The directive must be typed on one line, even though it is shown here on two lines.

purge
Remove access data files of a given age or when their collective size exceeds a given amount of storage.

none
Do not remove access data files. `none` is the default.

userexit
Specifies the path of the user-exit program you want to branch to. You can optionally specify the parameters for your user-exit program, as shown in the following examples. The server appends the path to the access log, to the directive.

Examples:

```
ReportDataArchive purge
ReportDataArchive none
ReportDataArchive userexit /u/userxyz/bin/backup/backup.rexx -d -a
```

For the `ReportDataArchive userexit` example, the user exit invocation is:

```
ReportDataArchive userexit /u/userxyz/bin/backup/backup.rexx
-d -a /www/logs/httpd-log
```

Note: The directive must be typed on one line, even though it is shown here on two lines.

Initial configuration file setting

```
ReportDataArchive none
```

Program default setting

ReportDataArchive none

ReportDataExpire - Remove existing access data files when they reach a given age in days

Use this directive to specify that you want to remove access log data files when they reach a certain age (in days).

This directive requires that you also specify the ReportDataArchive directive, described under "ReportDataArchive - Specify whether to remove existing accessdata files" on page 514. You can have only one occurrence of this directive in your configuration file.

```
ReportDataExpire   number-of-days
```

number-of-days
Specifies that reports older than this value are to be removed. *number-of-days* must be an integer; decimal values such as 1.5 are not valid. The default is 0, a value that indicates that no expiration date exists.

The file creation date, as reported by the operating system, is used to determine the age of the error log file. The suffix of the filename, such as `httpd-log.Mar221996`, is not used to determine file age.

Example

```
ReportDataExpire 10
```

Initial configuration file setting

```
ReportDataExpire 0
```

Program default setting

ReportDataExpire 0

SMF - Specify the type of information that SMF records

Use this directive to write configuration and performance data to a memory segment in preparation for analysis. The format of the directive is:

```
SMF settings
```

where *settings* can be **all**, **config**, **perf**, or **none**.

all Record both configuration and performance data to SMF.

config
Record only configuration data (record type 103, subtype 01) to SMF.

perf
Record only performance data (record type 103, subtype 02) to SMF.

none
Record no configuration or performance data to SMF.

Example

```
SMF perf
```

Initial configuration file setting

```
SMF none
```

Program default setting

```
SMF none
```

SMFRecordingInterval - Specify how often to record performance record information

Use the SMFRecordingInterval directive to define how often System Management Facility (SMF) writes performance record information to file. Performance record data accumulates continuously. See "Controlling the logging of information by SMF" on page 278 for exceptions to writing SMF performance records on an interval. The format of the directive follows:

```
SMFRecordingInterval hh:mm
```

hh:mm
Indicates the time that elapses between recordings of performance information.

Example

```
SMFRecordingInterval 00:20
```

The recording interval for performance information is 20 minutes.

Initial configuration file setting

```
SMFRecordingInterval 00:15
```

Program default setting

```
SMFRecordingInterval 00:15
```

Meta-Information - Name meta-information files and directories

Use the directives described in this section to control where your server looks for meta-information files.

You can use a separate set of files to store meta-information about your server's documents. The server can include the meta-information with its HTTP responses. Meta-information describes the file containing a document, not the contents of the document. For example, meta-information for a file might give the date the file was created and the date it was last modified. You can include any valid response headers as described in the HTTP 1.1 specification.

HTTP recognizes MIME headers. Information that MIME header fields can include are the file type, subtype, encoding, and content length.

Each line of a meta-information file contains a header field, followed by a colon, and the value of the field. For example:

```
Last-Modified: Wednesday, 05-Apr-96 20:51:35 GMT
Expires: Friday, 30-Jun-96 24:00:00 GMT
MIME-Version: 1.0
```

MetaDir - Specify name of subdirectory for meta-information files

Use this directive to specify the name you want to use for subdirectories that contain meta-information files. You can only have one instance of this directive, which means all your meta-information subdirectories have the same name. The stated directory is a subdirectory of the directory in the document which "meta" is located in.

Any directory from which your server retrieves files can have a subdirectory with the name specified on this directive. The files on the meta-information subdirectory contain meta-information about the files being retrieved. The meta-information files have the same file name and extension as the file they describe, plus an added extension. The name of the added suffix is specified on the MetaSuffix directive.

For example, you might have the following two directives in your configuration file:

```
MetaDir     look_here
MetaSuffix  .file_desc
```

If your server goes to retrieve this file:

```
/html/realcool/coolindex.html
```

it looks for meta information to include with the response in this file:

```
/html/realcool/look_here/coolindex.html.file_desc
```

Example

```
MetaDir  mimeinfo
```

Initial configuration file setting

```
MetaDir  .web
```

Note: The dot character (.) at the beginning of the default value is used as part of the subdirectory name.

Initial configuration file setting

```
MetaDir  .web
```

MetaSuffix - Specify the suffix for meta-information files

Use this directive to specify the suffix you want to use for meta-information files. You can only have one instance of this directive, which means all meta-information files end with the same suffix. You must include the period character (.) as part of the value.

Any file your server retrieves can have a meta-information file associated with it. A meta-information file has the same file name and suffix as the file it describes, plus the additional suffix specified on the MetaSuffix directive. The suffix is an extension of the complete ″meta″ file name. A meta-information file must be located on a subdirectory of the directory that contains the file being described. The name of the subdirectory must be the name specified on the MetaDir directive.

See the description of the MetaDir directive to see an example of how MetaDir and MetaSuffix work together.

Example

```
MetaSuffix   .head
```

Initial configuration file setting

```
MetaSuffix   .meta
```

Program default setting

```
MetaSuffix   .meta
```

Methods - Set method acceptance

Use the directives described in this section to control which HTTP methods are enabled for your server.

Client requests to the server include a method field that indicates the action the server is to perform on the specified object. The request identifies the object with a Uniform Resource Locator (URL).

Following is a list of methods that the server supports and a description of how the server would respond to a client request containing the method. The description assumes the method is enabled.

- CONNECT- This method is used to establish an SSL tunneling sessionbetween a client, such as NetScape Navigator, and a remote server through a proxy server. The sessions between the client and the proxy and between the proxy and the remote server are secure. The proxy does not have access to the data being sent. The proxy server can be a base or secure server.

- DELETE - The server deletes the object identified by the URL. After the object is deleted, the URL is not valid. Because delete typically lets clients delete information from your server, you must use protection setups to define who can use this method and which files can be deleted.
- GET - The server returns whatever data is identified by the URL.If the URL refers to an executable program, the server returns the output of the program.
- HEAD - The server returns only HTTP document headers without the document body.
- OPTIONS - The request returns information about the communicationsoptions on the request/response chain identified by the URL. This method allows a client to determine the options and requirements associated with an object, or the capabilities of a server, without having to act on or retrieve the object.
- POST - The request contains data and a URL.The server accepts the data enclosed in the request as a new subordinate of the resource identified in the URL. The resource, which may be a data-accepting program, a gateway to some other protocol, or a separate program that accepts annotations, processes the enclosed data. The POST method is designed to handle annotation of existing resources; posting of a message to a bulleting boards, newsgroup, mailing list, or similar group of articles; providing a block of data, such as data from a form to a data-handling program; or extending a database through an append operation. In the HTTP Server, the POST method is used to process the Configuration and Administration forms.
- PUT - The request contains data and a URL.The server stores the resource identified in the URL. If the resource already exists, PUT replaces it. If the resource does not exist, PUT creates it. Because PUT typically lets clients add or replace information on your server, you must use protection setups to define who can use this method for which files.
- TRACE - The server echos the request message sent by the client.This method allows the client to see what is being received at the other end of the request chain and use that data for testing or diagnostic information. The content type of the response is message/http.

Disable - Disable HTTP methods

Use this directive to specify which HTTP methods you do not want your server to accept.

In the default configuration file, the GET, HEAD, OPTIONS, POST, and TRACE methods are enabled and all other supported HTTP methods are disabled. To disable a method that is currently enabled, change the Enable directive for the method to a Disable directive.

Note: The Configuration and Administration forms use the POST method to make updates to your server configuration. If you disable the POST method you will not be able to use the Configuration and Administration forms.

Example

```
Disable HEAD
```

Initial configuration file setting

```
Disable   PUT
Disable   DELETE
```

Enable - Enable HTTP methods

Use this directive to specify which HTTP methods you want your server to accept.

Methods directives

You can enable as many of the HTTP methods as you need. For each method you want the server to accept, enter a separate Enable directive followed by the name of the method.

Example

```
Enable DELETE
```

If no Service directive exists for a particular URL, you can use the Enable directive to perform customized programming for any HTTP method. The program you specify on this directive will override the standard processing for that method.

The format is:

```
Enable     method /path/fileDLL:function_name
```

Example

```
Enable
```

Initial configuration file setting

```
Enable GET
Enable HEAD
Enable POST
Enable TRACE
Enable OPTIONS
```

Multi-format processing - Define file extensions for multi-format processing

Use the directives described in this section to associate files with particular extensions to the meta-information found in the headers of incoming requests.

Based on the file suffix specified in these directives, the server binds files to a content type, content encoding, content language, character set, or to a browser sending a request. A file suffix (or extension, or qualifier) is defined as a segment of a file name that is delimited by a dot (.) and contains the dot. The first segment of a file name is not considered a suffix.

Multi-Format Processing

Multi-format processing is only enabled when the requesting URL contains the .multi suffix or does not have a suffix (and a file with that name and no suffix does not exist).

csl1*

Multi-format processing includes:

- Request headers from the browser

  The browser sends accept-headers containing acceptable values (content-type, content-encoding, language, charset) that you can associate to file suffixes with the configuration directives. The browser also sends a user-agent header that identifies its browser type that you can associate with file suffixes in the same manner.
- The requested URL

  The server finds all files with any extension that matches the directory and file name and uses multi-format processing to choose the best file to return.

Summary of resource mapping directives and meta-information

The resource mapping directives, AddType, AddEncoding, AddLanguage, AddCharSet, and AddClient, are used to associate meta-information from request headers with file suffixes or extensions. Meta-information can consist of MIME type, encoding, quality, charset, language, and browser (agent) type. When processing a file for which a suffix or extension is not defined by an AddEncoding or AddType directive, the server uses the default value of binary for the content encoding.

The following list identifies meta-information that is associated with each directive:

- "AddType - Specify the data type of files with particular suffixes" on page 523
  - Suffix
  - MIME type
  - Encoding
  - Quality (optional)
  - Character set (optional)
- "AddEncoding - Specify the MIME content encoding of files with particular suffixes" on page 523
  - Suffix
  - Encoding
- "AddLanguage - Specify the language of files with particular suffixes" on page 522
  - Suffix
  - Language
- "AddCharSet - Specify the character set documents are encoded in" on page 523
  - Suffix
  - Character set
- "AddClient - Specify file extensions for requesting clients" on page 527
  - Suffix
  - User agent header

Computing file quality

Multi-format processing computes the quality of a file based on the set of suffixes in the directory for the requested file name and sends the highest quality it finds. A perfect match is always the highest quality.

Quality is a floating point number between 0.0 and 1.0 that represents the relative desirability of a file. For example, if the file abc.html is requested and cannot be found, the server searches the directory for all files that match abc.*. Multi-format processing adds the * and finds all that may qualify.

When computing file quality, each file is given the value of 1.0 and this is multiplied by the quality of each suffix. The quality of a suffix is based on the quality value specified in the directive which is multiplied by the quality value specified in the accept-header. This can be shown as:

```
fileQ = (1.0 X ((suffixQ1) X (suffixQ2)...))
```

or

```
fileQ = (1.0 X ((directiveQ X headerQ) X (directiveQ X headerQ)...))
```

The quality value is optional on many of the directives. If one is not specified, then the value of 1.0 is assumed. Also, if the accept-header does not specify a quality value, 1.0 is assumed.

If the server is processing a file and suffix that is defined in the list of AddClient directives, and the requester agent matches the defined string, the server doubles the file quality (implying a value of 2.0). If no match is found, this does not happen.

Example: If you have HTML source files in different languages,

```
myfile.du.html
myfile.uk.html
```

you could use these directives for multi-format processing:

```
AddLanguage .du du
AddLanguage .uk en_UK
AddType .html text/html ebcdic
```

Then, if a browser sends a request for `myfile.multi` and sends the header `Accept-Language: du`, the server returns myfile.du.html. (Both files are stored in EBCDIC charset IBM-1047 and translated to ASCII charset ISO8859-1 when served.) The configuration file shipped with the server contains specifications for most commonly used suffixes. Use the suffix definition directives only if you need to add new definitions or change the sample definitions found in the configuration file.

AddLanguage - Specify the language of files with particular suffixes

Use this directive to bind files with a particular suffix to a language. The format of the directive is:

AddLanguage *.extension language*

.extension
The file suffix pattern.

language
The language you want to bind to files that match the corresponding suffix pattern.

Examples

```
AddLanguage .en  en_US
```

This example defines files with a .en suffix as being in American English.

```
AddLanguage .uk  en_UK
```

This example defines files with a .uk suffix as being in United Kingdom English.

Initial configuration file setting

None.

Program default setting

None.

AddEncoding - Specify the MIME content encoding of files with particular suffixes

Use this directive to bind files with a particular suffix to a MIME-encoding type.

The format of the directive is:

```
AddEncoding  .extension encoding
```

.extension
The file suffix pattern.

encoding
The MIME-encoding type you want to bind to files that match the corresponding suffix pattern.

Example

```
AddEncoding .qp   quoted_printable
```

Initial configuration file setting

```
AddEncoding .Z       x-compress
AddEncoding .gz      x-gzip
AddEncoding .ascii   8bit
AddEncoding .asc8    8bit
AddEncoding .asc7    7bit
AddEncoding .ebcdic  ebcdic
```

AddCharSet - Specify the character set documents are encoded in

Use this directive to specify the character set (code page) documents are stored in. (You can also specify character set on the AddType directive.)

The format of the directive is:

```
AddCharSet  .extension character-set
```

.extension
The file suffix pattern.

character-set
The character set you want to associate with text documents. This should indicate the character set transmitted. If encoding is specified as ebcdic, the character set is ISO8859-1. For the documents that you assign a character set to, the server tells client browsers what character set to use when displaying the document. If you want to use the character-set field, you must also include a value for the quality field. If you use DefaultFsCp or DefaultNetCp, you might want to use *character-set* to indicate displayed or stored binary pages.

Example

```
AddCharSet .932   IBM-932
```

Initial configuration file setting

None.

AddType - Specify the data type of files with particular suffixes

Use this directive to bind files with a particular suffix to a MIME type/subtype. You can have multiple occurrences of this directive in your configuration file. The format of the directive is:

```
AddType  .extension type/subtype encoding [quality[ character-set]]
```

.extension
The file suffix pattern. You can use the wildcard character (*) only on the following two special suffix patterns:

.
Matches all file names that contain a dot character (.) and have not been matched by other rules

*****
Matches all file names that do not contain a dot character (.) and have not been matched by other rules

type/subtype
The MIME type and subtype you want to bind to files that match the corresponding suffix pattern.

encoding
The MIME content encoding to which the data has been converted. On MVS, text data is transmitted in 7bit or 8bit ASCII and stored in the HFS as EBCDIC. The iconv() service is used to translate between ASCII codepage ISO8859-1 and EBCDIC codepage IBM-1047. Only files with ebcdic encoding are translated.

Possible values of *encoding* are:

7bit
All data is represented as short lines (less than 1000 characters) of 8859-1 ASCII data. Source code or plain text files usually fall into this category. Exceptions would be files containing line-drawing characters or accented characters.

8bit
Data is represented as short lines, but may contain characters with the high bit set (for example, line-drawing characters or accented characters). PostScript files and text files from European sites usually fall into this category.

binary
This encoding can be used for all data types. Data may contain not only non-ASCII characters, but also long (greater than 1000 characters) lines. Almost every file of type image/*, audio/*, and video/* falls into this category, as do binary data files of type application/*.

ebcdic
Data is represented as lines of EBCDIC text (IBM-1047) and data is translated to ASCII text (ISO8859-1) when served. Note that codepages can be overridden by the DefaultFsCp and DefaultNetCp directives.

other
Any *other* value of encoding receives the same treatment as binary.

If *encoding* is `ebcdic`, the server will translate the content to ASCII characters. If *encoding* is any other value, the server will not translate the content.

If *encoding* is `7bit`, `8bit`, `binary` or `ebcdic`, the server will not forward the Content-Encoding header to the browser.

quality
An optional indicator of relative value (on a scale of 0.0 to 1.0) for the content type. The quality value is used if multiple representations of a file are matched by a request. The server selects the file that is associated with the highest quality value. For example, if the file internet.ps is requested, and the server has the following AddType directives:

```
AddType   .ps application/postscript   8bit   1.0
AddType   *.* application/binary       binary 0.3
```

the server would use the application/postscript line because its quality number is higher.

character-set

An optional indicator of the character set you want to associate with text documents. This should indicate the character set transmitted. If encoding is specified as ebcdic, the character set is ISO8859-1. For the documents that you assign a character set to, the server tells client browsers what character set to use when displaying the document. If you want to use the character-set field, you must also include a value for the quality field. If you use DefaultFsCp or DefaultNetCp, you might want to use *character-set* to indicate displayed or stored binary pages.

You can also use the AddCharSet directive to specify character set.

Example

```
AddType  .html  text/html  ebcdic  1.0
```

Initial configuration file settings

Note: Many of the AddType directives below appear on two lines for printing purposes. Each of these directives appears on one line in your log.

```
AddType  .cer      application/x-x509-user-cert  ebcdic  0.5 # Browser Certificate
AddType  .der      application/x-x509-ca-cert    binary  1.0 # CA Certificate
AddType  .mime     www/mime                      binary  1.0 # Internal
AddType  .bin      application/octet-stream      binary  1.0 # Uninterpreted binary
AddType  .class    application/octet-stream      binary  1.0 # Java™ applet
                                                             # or application
#AddType .oda      application/oda               binary  1.0
AddType  .pdf      application/pdf               binary  1.0
AddType  .ai       application/postscript        ebcdic  0.5 # Adobe Illustrator
AddType  .PS       application/postscript        ebcdic  0.8 # PostScript
AddType  .eps      application/postscript        ebcdic  0.8
AddType  .ps       application/postscript        ebcdic  0.8
AddType  .rtf      application/x-rtf             ebcdic  1.0 # RTF
AddType  .csh      application/x-csh             ebcdic  0.5 # C-shell script
#AddType .dvi      application/x-dvi             binary  1.0 # TeX DVI
#AddType .hdf      application/x-hdf             binary  1.0 # NCSA HDF data file
AddType  .latex    application/x-latex           ebcdic  1.0 # LaTeX source
#AddType .nc       application/x-netcdf          binary  1.0 # Unidata netCDF data
AddType  .cdf      application/x-cdf             ebcdic  1.0 # Channel Definition
                                                             # Format
AddType  .sh       application/x-sh              ebcdic  0.5 # Shell-script
AddType  .tcl      application/x-tcl             ebcdic  0.5 # TCL-script
AddType  .tex      application/x-tex             ebcdic  1.0 # TeX source
#AddType .texi     application/x-texinfo         ebcdic  1.0 # Texinfo
#AddType .texinfo  application/x-texinfo         ebcdic  1.0
AddType  .t        application/x-troff           ebcdic  0.5 # Troff
AddType  .roff     application/x-troff           ebcdic  0.5
AddType  .tr       application/x-troff           ebcdic  0.5
AddType  .man      application/x-troff-man       ebcdic  0.5 # Troff with man macros
AddType  .me       application/x-troff-me        ebcdic  0.5 # Troff with me macros
AddType  .ms       application/x-troff-ms        ebcdic  0.5 # Troff with ms macros
#AddType .src      application/x-wais-source     ebcdic  1.0 # WAIS source
#AddType .bcpio    application/x-bcpio           binary  1.0 # Old binary CPIO
#AddType .cpio     application/x-cpio            binary  1.0 # POSIX CPIO
AddType  .gtar     application/x-gtar            binary  1.0 # Gnu tar
AddType  .shar     application/x-shar            ebcdic  1.0 # Shell archive
#AddType .sv4cpio  application/x-sv4cpio         binary  1.0 # SVR4 CPIO
#AddType .sv4crc   application/x-sv4crc          binary  1.0 # SVR4 CPIO with CRC
AddType  .wrl      x-world/x-vrml                binary  1.0 # VRML
```

The following are neutral CAE formats:

```
#AddType .igs      application/iges              binary  1.0 # IGES Graphics format
#AddType .iges     application/iges              binary  1.0 # IGES Graphics format
#AddType .IGS      application/iges              binary  1.0 # IGES Graphics format
```

```
#AddType .IGES     application/iges              binary  1.0 # IGES Graphics format
#AddType .stp      application/STEP              ebcdic  1.0 # ISO-10303 STEP  -
#AddType .STP      application/STEP              ebcdic  1.0 # Product data files
#AddType .step     application/STEP              ebcdic  1.0
#AddType .STEP     application/STEP              ebcdic  1.0
#AddType .dxf      application/dxf               binary  1.0 # DXF (AUTODESK)
#AddType .DXF      application/dxf               binary  1.0
#AddType .vda      application/vda               binary  1.0 # VDA-FS Surface data
#AddType .VDA      application/vda               binary  1.0
#AddType .set      application/set               ebcdic  1.0 # SET(French CAD
                                                             # standard)
#AddType .SET      application/set               ebcdic  1.0
#AddType .stl      application/SLA               ebcdic  1.0 #  Stereolithography
#AddType .STL      application/SLA               ebcdic  1.0
```

The following are vendor-specific CAD-formats commonly used by CERN and in HEP institutes:

```
#AddType .dwg      application/acad              binary  1.0 # Autocad drawing files
#AddType .DWG      application/acad              binary  1.0
#AddType .SOL      application/solids            binary  1.0 # MATRA Prelude solids
#AddType .DRW      application/drafting          binary  1.0 # Prelude Drafting
#AddType .prt      application/pro_eng           binary  1.0 # PTC Pro/ENGINEER part
#AddType .PRT      application/pro_eng           binary  1.0
#AddType .unv      application/i-deas            binary  1.0 # SDRC I-DEAS files
#AddType .UNV      application/i-deas            binary  1.0
#AddType .CCAD     application/clariscad         binary  1.0 # ClarisCAD files
AddType  .snd      audio/basic                   binary  1.0 # Audio
AddType  .au       audio/basic                   binary  1.0
AddType  .aiff     audio/x-aiff                  binary  1.0
AddType  .aifc     audio/x-aiff                  binary  1.0
AddType  .aif      audio/x-aiff                  binary  1.0
AddType  .wav      audio/x-wav                   binary  1.0 # Windows+ WAVE format
AddType  .bmp      image/bmp                     binary  1.0 # OS/2 bitmap format
AddType  .gif      image/gif                     binary  1.0 # GIF
AddType  .ief      image/ief                     binary  1.0 # Image Exchange format
AddType  .jpg      image/jpeg                    binary  1.0 # JPEG
AddType  .JPG      image/jpeg                    binary  1.0
AddType  .JPE      image/jpeg                    binary  1.0
AddType  .jpe      image/jpeg                    binary  1.0
AddType  .JPEG     image/jpeg                    binary  1.0
AddType  .jpeg     image/jpeg                    binary  1.0
AddType  .tif      image/tiff                    binary  1.0 # TIFF
AddType  .tiff     image/tiff                    binary  1.0
AddType  .ras      image/cmu-raster              binary  1.0
AddType  .pnm      image/x-portable-anymap       binary  1.0 # PBM Anymap format
AddType  .pbm      image/x-portable-bitmap       binary  1.0 # PBM Bitmap format
AddType  .pgm      image/x-portable-graymap      binary  1.0 # PBM Graymap format
AddType  .ppm      image/x-portable-pixmap       binary  1.0 # PBM Pixmap format
AddType  .rgb      image/x-rgb                   binary  1.0
AddType  .xbm      image/x-xbitmap               ebcdic  1.0 # X bitmap
AddType  .xpm      image/x-xpixmap               binary  1.0 # X pixmap format
AddType  .xwd      image/x-xwindowdump           binary  1.0 # X window dump (xwd)
AddType  .html     text/html                     ebcdic  1.0 # HTML
AddType  .htm      text/html                     ebcdic  1.0 # HTML on PCs
AddType  .htmls    text/x-ssi-html               ebcdic  1.0 # Server-side includes
AddType  .shtml    text/x-ssi-html               ebcdic  1.0 # Server-side includes
AddType  .c        text/plain                    ebcdic  0.5 # C source
AddType  .h        text/plain                    ebcdic  0.5 # C headers
AddType  .C        text/plain                    ebcdic  0.5 # C++ source
AddType  .cc       text/plain                    ebcdic  0.5 # C++ source
AddType  .hh       text/plain                    ebcdic  0.5 # C++ headers
AddType  .java     text/plain                    ebcdic  0.5 # Java source
AddType  .m        text/plain                    ebcdic  0.5 # Objective-C source
AddType  .f90      text/plain                    ebcdic  0.5 # Fortran 90 source
AddType  .txt      text/plain                    ebcdic  0.5 # Plain text
```

```
AddType  .css      text/css                     8bit    1.0 # W3C Cascading Style
                                                            # Sheets
AddType  .rtx      text/richtext                ebcdic  1.0 # MIME Richtext format
AddType  .tsv      text/tab-separated-values    ebcdic  1.0 # Tab-separated values
AddType  .etx      text/x-setext                ebcdic  0.9 # Structured Enhanced
                                                            # Text
AddType  .MPG      video/mpeg                   binary  1.0 # MPEG
AddType  .mpg      video/mpeg                   binary  1.0
AddType  .MPE      video/mpeg                   binary  1.0
AddType  .mpe      video/mpeg                   binary  1.0
AddType  .MPEG     video/mpeg                   binary  1.0
AddType  .mpeg     video/mpeg                   binary  1.0
AddType  .qt       video/quicktime              binary  1.0 # QuickTime
AddType  .mov      video/quicktime              binary  1.0
AddType  .avi      video/x-msvideo              binary  1.0 # MS Video for
                                                            # Windows
AddType  .movie    video/x-sgi-movie            binary  1.0 # SGI  movieplayer
AddType  .zip      multipart/x-zip              binary  1.0 # PKZIP
AddType  .Z        application/x-compress       gzip    1.0 # PKZIP
AddType  .gz       application/x-compress       gzip    1.0 # PKZIP
AddType  .tar      multipart/x-tar              binary  1.0 # 4.3BSD tar
AddType  .ustar    multipart/x-ustar            binary  1.0 # POSIX tar
AddType  *.*       www/unknown                  binary  0.2
AddType  *         www/unknown                  binary  0.2
AddType  .cxx      text/plain                   ebcdic  0.5 # C++
AddType  .for      text/plain                   ebcdic  0.5 # Fortran
AddType  .mar      text/plain                   ebcdic  0.5 # MACRO
AddType  .log      text/plain                   ebcdic  0.5 # logfiles
AddType  .com      text/plain                   ebcdic  0.5 # scripts
AddType  .sdml     text/plain                   ebcdic  0.5 # SDML
AddType  .list     text/plain                   ebcdic  0.5 # listfiles
AddType  .lst      text/plain                   ebcdic  0.5 # listfiles
AddType  .def      text/plain                   ebcdic  0.5 # definition files
AddType  .conf     text/plain                   ebcdic  0.5 # definition files
AddType  .         text/plain                   ebcdic  0.5 # files with no
                                                            # extension
```

The following can be used to store ASCII DBCS:

```
AddType  .JP932    text/x-DBCS                  binary  1.0 IBM-932  # Japanese
                                                                     # DBCS
AddType  .JPeuc    text/x-DBCS                  binary  1.0 IBMeucJP # Japanese
                                                                     # DBCS
```

AddClient - Specify file extensions for requesting clients

Use this directive to bind files with particular extensions to the type and version of client that is sending the request. This is often referred to as ***automatic browser detection***.

All HTTP requests contain a `User-Agent` header that identifies the browser sending the request. The HTTP Server enables you to detect which browser was used to send a request and, based on this information, respond with a version of a Web page, a document, or other file that is appropriate for that browser.

For example, your server can send a page written in HTML 3.0 only to browsers that are known to support it and send a version of the same page written in HTML 2.0 to all other browsers.

Automatic browser detection is only effective for multi-format processing. If you use multi-format processing, a requesting URL specifies a file without an extension and no file with that name exists, or a requesting URL specifies a file with the **.multi** extension. For example, a link from this HTML anchor tag initiates multi-format processing:

```
<A HREF="http://www.mycompany.com/mydept/tscores.multi">
```

As a result, the server will evaluate the values passed in the request headers, along with the extensions of all the tscores files and the associations specified in the directives. Based on this, it will try to find the file that is the best match to send in its response.

You can have multiple occurrences of this directive in your configuration file. The sequence of AddClient directives is important. The first AddClient directive that matches a client's User-Agent value is the one that will be used to determine the file extension.

If a client's User-Agent is not matched in an AddClient directive, the server looks for a generic file extension (.htm or .html) to send. If the server cannot find a generic file extension, it uses an algorithm to calculate the quality of all the extensions for that file and sends the file whose extension yields the highest quality, considering it to be the best match.

The format of the directive is:

```
AddClient .extension user-agent-header
```

.extension
: The file extension you specify for the file you want to send to a particular browser.

  This extension can be one of a string of suffixes used to qualify a file. For example, the extension .Netscape can apply to a file named TxtSample.UK.Netscape.html or TxtSample.html.Netscape.eng. You cannot use any wildcard characters in the file extension.

  This extension may also be defined on an AddType or other multi-format processing directive, to further define the attributes of the file containing this extension.

user-agent-header
: This field must match the value in the User-Agent header of the incoming request.

  This field is case-sensitive. You can use an asterisk (*) as a wildcard character in this field or include the complete header information. Use quotes to include white space in the user-agent name.

  Examples:

  - `Mozilla/2.*` applies to all levels of Version 2 of Netscape's browser, Netscape Navigator.
  - `"Mozilla/4.0 (compatible; MSIE)"` applies to Microsoft's browser, Internet Explorer Version 4.0.

Examples

```
AddClient  .Netscape      Mozilla/2.*
AddClient  .MSIE          "Mozilla/4.0 (compatible; MSIE)"
```

Initial configuration file setting

None.

Program default setting

None.

Using automatic browser detection for Welcome pages

This example describes how to enable automatic browser detection for Welcome pages. It shows how to serve different versions of the index.html file in the webhome directory. You can also use this example for Welcome.html, welcome.html, or Frntpage.html files.

Follow these steps to enable browser detection and multi-format processing for your welcome pages:

1. Add a Welcome directive to your configuration file that specifies the file name with the .multi extension:

   ```
   Welcome  index.multi
   ```

2. Include AddClient directives in the configuration file that specify which file extensions to send to a particular browser. You can use an asterisk (*) as a wildcard character or include the complete header information. Use quotes to include white space in the user-agent name.

   Examples:

   ```
   AddClient  .Netscape      Mozilla/2.*
   AddClient  .MSIE          "Mozilla/4.0 (compatible; MSIE)"
   ```

3. Create specific versions of the index file for each of these file extensions: index.Netscape.html and index.MSIE.html.
4. Create a dummy file in the directory called index.multi.
5. Specify only the directory name in the URL when linking to this page.

   ```
   http://www.web4hire.com/webhome/
   ```

SuffixCaseSense - Specify whether suffix definitions are case sensitive

Use this directive to specify whether you want your server to distinguish between uppercase and lowercase letters when comparing file suffixes to the suffix patterns on AddClient, AddCharSet, AddType, AddEncoding, and AddLanguage directives. By default, the server does not distinguish between uppercase and lowercase.

Example

```
SuffixCaseSense  On
```

Initial configuration file setting

```
SuffixCaseSense  Off
```

Program default setting

```
SuffixCaseSense  Off
```

Proxy server settings — Configure the Web server as a proxy server

Use the directives in this section to configure a proxy server. For more information on using IBM Web Traffic Express to set up a proxy server, see Chapter 16, “Running your server as a proxy” on page 303.

CacheDefaultExpiry - Specify a default expiration date

Use this directive to specify the expiration date for files which do not include an explicit expiration date or a last-modified date that enables the server to compute an expiration date based on the setting of the CacheLastModifiedFactor directive. This is most useful for protocols that do not have any way to transmit this information, such as FTP or Gopher.

Initial configuration file setting

```
CacheDefaultExpiry http:*  0 days
CacheDefaultExpiry ftp:*  1 day
CacheDefaultExpiry gopher:*  2 days
```

Program default setting

```
CacheDefaultExpiry http:*  0 days
CacheDefaultExpiry ftp:*  1 day
CacheDefaultExpiry gopher:*  2 days
```

It is recommended that the HTTP default of 0 days be used.

CacheExpiryCheck - Specify whether the server will return expired files

Use this directive to specify whether you want the server to return cached files that have expired. Normally, a caching proxy will check that the files in its cache have not expired. In special circumstances, such as a network outage, you might want to disable this check; the proxy will then serve files from its cache even if they are out of date. Generally, you do not want the server to return expired files.

With the default value of `On`, the server will not return expired files. Specify `Off` if you want the server to be able to return expired files.

Initial configuration file setting

```
CacheExpiryCheck On
```

Program default setting

```
CacheExpiryCheck On
```

CacheLastModifiedFactor - Specify fraction of Last-Modified date to use in determining expiration date

Use this directive if you want the server to set expiration dates for files which have a Last-Modified date but no Expiration date. The server uses the Last-Modified date to determine how long it has been since the file was modified, then multiplies that time by the fraction specified on this directive to determine the expiration date. For example, if the file was changed 2 days ago, and the CacheLastModifiedFactor was set to 0.5, the file would expire in one day (2 multiplied by 0.5).

Files that change more recently are refreshed sooner. The assumption is that files that have changed recently are probably changing frequently whereas files that have not been modified in some time do not need to be refreshed as frequently.

Initial configuration file setting

```
CacheLastModifiedFactor 0.14
```

Program default setting

```
CacheLastModifiedFactor 0.14
```

CacheLimit_2 - Specify upper limit for cached file size

Use this directive to specify the maximum size of files to be cached. Files larger than this size will not be cached. This value can be specified in bytes (B), kilobytes (K), megabytes (M), or gigabytes (G).

Initial configuration file setting

```
CacheLimit_2 4000 K
```

Program default setting

```
CacheLimit_2 4000 K
```

CacheNoConnect - Specify stand alone cache mode

Use this directive to specify whether you want the proxy server to retrieve files from remote servers. Use the default value of `Off` if you want the server to be able to retrieve files from remote servers.

Specify `On` if you want the server to run in stand alone cache mode. This means that the server can return only files already stored in its cache. Typically, you would also set the CacheExpiryCheck directive to `Off` when running the server in this mode.

Running the server in stand alone cache mode can be useful if you are using the server for demonstrations. If you know all the files you want to use for a demonstration are stored in the cache, then you do not need a network connection.

Initial configuration file setting

```
CacheNoConnect Off
```

Program default setting

```
CacheNoConnect Off
```

CacheOnly - Cache only files with URIs that match a template

Use this directive to specify that only files with URIs that match the given template should be cached. You can have multiple occurrences of this directive in the configuration file. Include a separate directive for each template. The URI template must include the protocol.

Example

```
CacheOnly http://www.ibm.com/*
```

Initial configuration file setting

None

Program default setting

None

CacheRoot - Specify cache root directory

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the directory that the proxy server will use for caching files. This directory is required if the Caching directive is set on.

If this directive is specified using the Configuration and Administration Forms, the directory will be created automatically. If you define the directory by manually

editing the configuration file, you need to create the directory with the appropriate permissions to enable the server to cache documents in it.

Example

```
CacheRoot /webcache
```

Initial configuration file setting

None

Program default setting

None

CacheSize - Specify cache size

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the size of the proxy server's cache in megabytes. When the maximum size is reached, the garbage collection process begins.

Initial configuration file setting

```
CacheSize 5 M
```

Program default setting

```
CacheSize 5 M
```

CacheUnused - Specify how long to keep unused cached files that match a template

Use this directive to set the maximum amount of time for the server to keep unused cached files with URIs matching a given template. The server deletes unused files with URIs matching the template after they have been cached for the specified time, regardless of their expiration date.

You can have multiple occurrences of this directive in the configuration file. Include a separate directive for each template. The URI template must include the protocol. Specify the time value in any combination of months, weeks, days, and hours.

Examples

```
CacheUnused ftp:* 1 day
CacheUnused gopher:* 3 days 12 hours
CacheUnused http:* 2 days
```

Initial configuration file setting

None

Program default setting

None

Caching - Turn proxy caching off or on

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to turn proxy caching on or off. With caching turned on, the proxy server can store the files it retrieves from other servers in a local cache. The server can then respond to subsequent requests for the same files without having to retrieve them from other servers. This can improve response time.

If Caching is set on, you must also specify the CacheRoot directive.

Initial configuration file setting

```
Caching Off
```

Program default setting

```
Caching Off
```

ftp_proxy - Specify a proxy server for this proxy to connect to for FTP requests

If your proxy server is part of a chain of proxies, use this directive to specify the name of another proxy that this server should contact for FTP requests.

Example

```
ftp_proxy http://other.proxy.server/
```

Initial configuration file setting

None

Program default setting

None

Gc - Turn garbage collection on or off

If you have enabled caching, the server uses the garbage collection process to delete files that should no longer be cached. Files are deleted based on their expiration date and other proxy directive values. Use this directive to turn garbage collection on or off. Generally, you would not turn off garbage collection if you have enabled caching. If you do, your cache file could grow beyond the maximum size you set for it.

When garbage collection is turned on, the garbage collection process runs when the cache reaches its maximum size (as specified on the CacheSize directive). The garbage collection process will also run at the time of day specified on the GcDailyGc directive.

Initial configuration file setting

```
Gc Off
```

Program default setting

```
Gc Off
```

GcDailyGc - Specify a daily time for garbage collection

Use this directive to specify a particular time of day to run the garbage collection process. Garbage collection occurs automatically when the cache size limit is reached. By specifying a daily time for garbage collection you can also remove cached files before the cache reaches its maximum. Specify the time value in 24:00 hour format. Generally, you would want the garbage collection process to run when your server is not being used much for other things. This is why the default is 03:00.

Example

```
GcDailyGc 22:00
```

The above example would start the garbage collection process at 10 P.M.

Initial configuration file setting

```
GcDailyGc 03:00
```

Program default setting

```
GcDailyGc 03:00
```

GcMemUsage - Specify how much memory to use for garbage collection

Garbage collection works best if it can read all cache information into memory at one time. It may not be able to read in the entire cache if your system does not have enough main memory.

Use this directive to specify how much memory garbage collection can use. The value you specify should be approximately the amount of virtual memory that the server may use while performing garbage collection. The amount of memory needed will vary based on dynamic changes such as the directory structure of cached files. Specify the value as a number that represents kilobytes. The minimum value is 20.

Initial configuration file setting

```
GcMemUsage 500
```

Program default setting

```
GcMemUsage 500
```

gopher_proxy - Specify a proxy server for this proxy to connect to for Gopher requests

If your proxy server is part of a chain of proxies, use this directive to specify the name of another proxy that this server should contact for Gopher requests.

Example

```
gopher_proxy gopher://other.proxy.server/
```

Initial configuration file setting

None

Program default setting

None

http_proxy - Specify a proxy server for this proxy to connect to for HTTP requests

If your proxy server is part of a chain of proxies, use this directive to specify the name of another proxy that this server should contact for HTTP requests.

Example

```
http_proxy http://other.proxy.server/
```

Initial configuration file setting

None

Program default setting

None

no_proxy - Connect directly to domains matching templates

Use this directive to specify the domains that you want the Web server to directly connect to rather than going through a proxy.

Specify the value as a string of domain names or domain name templates. Separate each entry in the string with a comma. Do **not** put any spaces in the string.

You specify templates on this directive differently than the way you specify templates on other directives. Most importantly, you **cannot** use the wildcard character (*). What you **can** do is specify a template by including only the last part of a domain name. The server connects directly to any domains that end with a string matching the templates you specify. You can specify IP addresses and match the beginning portion of the string.

Example

```
no_proxy   www.someco.com,.raleigh.ibm.com,.some.host.org:8080
```

In the above example the server would not go through a proxy for the following requests:

- Any requests to domains ending with www.someco.com
- Any requests to domains ending with .raleigh.ibm.com, such as blugrass.raleigh.ibm.com or keystone.raleigh.ibm.com
- Any requests to port 8080 of domains ending with .some.host.org, such as myname.some.host.org:8080. This would not include requests to any other ports of the same domain, such as myname.some.host.org, which assumes the default port 80.

Initial configuration file setting

None

Program default setting

None

NoCaching - Do not cache files with URIs that match a template

Use this directive to specify that the Web server should not cache files with URIs matching the given template. You can have multiple occurrences of this directive in the configuration file. Include a separate directive for each template.

Example

```
NoCaching  http://never.cache.me.net/*
```

Initial configuration file setting

None.

Program default setting

None

ProxyAccessLog — Name the path for the proxy access log file

For information on ProxyAccessLog, see "Logging and Reporting - Customize logs and generate reports" on page 491.

SocksServer - Specify a socks server

Use this directive to specify the IP address or host name of the socks server through which this proxy will be passing requests.

Format

```
SocksServer SocksServer
```

where *SocksServer* is the IP address or the host name of the Socks server to which the proxy should be chained.

Examples

```
SocksServer 9.37.134.8
SocksServer socks.bcd.com
```

Initial configuration file setting

```
None
```

Program default setting

```
None
```

Resource mapping - Redirect URLs

Use the directives described in this section to control which requests your server accepts and where the server looks for resources.

Use the mapping directives (Exec, Fail, Map, Pass, and Redirect) to control which requests your server accepts and to map URL requests to your actual files.

You can use the mapping directives to create a virtual hierarchy of Web resources. You can then change the physical location of files or directories without affecting the virtual layout. Even if your server sends documents from different file systems, it can present a virtual layout.

The mapping directives allow the administrator to create a virtual hierarchy of Web resources that is independent of their physical location in the file hierarchy. Careful design of a virtual hierarchy will allow changes in the physical location of Web resources without impact to the virtual hierarchy. As a result, the physical file structure of a Web site can be hidden from untrusted users. Through the virtual hierarchy, mapping directives tell the Web server what requests to accept, who can request them, and where the requested Web resources can be found in the physical file system.

The order of the mapping directives in the configuration file is very important. The server applies the mapping directives in the order they appear in the configuration file until a request has been accepted, rejected, or there are no more directives that apply to the request. When constructing the virtual hierarchy be careful to specify specific mappings before more generic mappings. For example:

```
EXEC   /Zcgi-bin/RAR/* /usr/lpp/internet/server_root/ZCgi-bin/RAR/*
```

is more specific than the rule

```
EXEC   /Zcgi-bin/*   /usr/lpp/internet/server_root/ZCgi-bin/*
```

which is more specific than the rule

```
PASS   /*           /usr/lpp/internet/server_root/ZPages/*
```

The request

```
http://mvsesa06.tcp.com/Zcgi-bin/calendar.sh
```

is mapped to

```
/usr/lpp/internet/server_root/ZCgi-bin/calendar.sh
```

If the order of the rules are reversed, the request:

```
http://mvsesa06.tcp.com/Zcgi-bin/calendar.sh
```

is mapped to

```
/usr/lpp/internet/server_root/ZPages/Zcgi-bin/calendar.sh
```

This is not the desired mapping.

Exec - Run a CGI program for matching requests

Use this directive to specify a template for requests you want to accept and respond to by running a CGI program. Once a request matches a template on an Exec directive, the request is not compared to request templates on any subsequent directives.

The format of the directive is:

```
Exec  request-template program-path [Server-IP-address or hostname]
```

request-template
: A template for requests that you want your server to accept and respond to by running a CGI program.

  You must use an asterisk as a wildcard in both the *request-template* and *program-path*. The part of the request that matches the *request-template* wildcard must begin with the name of the file that contains the CGI program. The wildcard is optional, but if used, it must be in both the *request-template* and *program-path*.

  The request can also contain additional data that is passed to the CGI program in the PATH_INFO environment variable. The additional data follows the first slash character that comes after the CGI program file name on the request. The data is passed according to CGI specifications.

program-path
: The path to the file that contains the CGI program you want the server to execute for the request. *program-path* must also contain a wildcard. The

wildcard is replaced with the name of the file that contains the CGI program. The wildcard is optional, but if used, it must be in both the *request-template* and *program-path*.

The Exec directive is recursive and applies to all subdirectories. You do not need a separate Exec directive for each directory under cgi-bin and admin-bin.

Server-IP-address **or** *hostname*
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. (For more information on using multiple IP addresses or virtual hosts, see Chapter 17, "Running your server with multiple IP addresses or virtual hosts" on page 309.) The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

You can specify an IP address (for example, 204.146.167.72) or you can specify a host name (for example, hostA.bcd.com).

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

A wildcard character cannot be specified for a server's IP address.

Note: If a user requests a URL that matches an Exec directive and the resulting path specifies a directory, the server either fails the request with a 404 error code, or treats this situation as if a Pass directive were matched instead, depending on the setting of the ExecDirPass directive. See the ExecDirPass directive for more infomation.

If an Exec directive results in a Java™ CGI being executed, the server invokes the Java™ Virtual Machine, passing it the name of the class file. The Java Virtual Machine finds the file in the list of directories specified in the CLASSPATH environment variable and executes the first instance of the file in this list. This may not be the same path that is specified in the program path value.

For more information on how to set up class files for use as Java CGI's, see Chapter 18, "Writing Common Gateway Interface programs" on page 315.

Examples

```
Exec  /idd/depts/*  /depts/bin/*
```

In this example, if your server receives a request of /idd/depts/plan/c92, it attempts to run a CGI program or script matching the following name:

```
/depts/bin/plan
```

In the example, c92 is then passed to the first matching CGI program or script as input.

Initial configuration file setting

```
Exec  /cgi-bin/*      /usr/lpp/internet/server_root/cgi-bin/*
Exec  /admin-bin/*    /usr/lpp/internet/server_root/admin-bin/*
```

The HTTP Server supports CERN preparsed server scripts; however, it is recommended that they be upgraded to CGI scripts. Preparsed server scripts must end with the suffix of .pp when being referenced in URLs. The text following the

script name in the URL is always treated by the server as the first parameter to the script. For more information about CERN preparsed server scripts, go to URL:

```
http://www.w3.org/pub/WWW/Daemon/User/HTBinDoc.html
```

ExecDirPass - Control access to directories matching Exec directives

Use this directive to specify whether you want your server to allow implicit access to directory names matching Exec directives. The purpose of the Exec directive is to have the server execute a CGI, but if a URL is requested that resolves to only a directory name, this is obviously not a CGI, so it cannot be executed. If you want to let users access directories in this way, use:

```
ExecDirPass On
```

The server treats the Exec directive as if it is a Pass directive, and returns a directory listing or welcome page, if one is available. See the DirAccess and AlwaysWelcome directives for more information on how passed directories are handled.

If you specify ExecDirPass On, you do not need to specify separate Pass statements to allow directory listings of CGI directories. If you do not want to implicitly allow directory access to all directories matching Exec directives, then you must still specify Pass directives for those that you want to allow access to.

Note: Use care when deciding to use this directive. If you enable this function, the names of all files (including CGIs) will become visible when browsing the resulting directory. This may give end users insight into your system conventions, which may expose sensitive information.

If you do not specify this directive, or if you specify Off, the server returns a 404 Failed By Rule error.

Examples

```
ExecDirPass On
ExecDirPass Off
```

Program default setting

```
ExecDirPass Off
```

Fail - Reject matching requests

Use this directive to specify a template for requests you do not want to process. Once a request matches a template on a Fail directive, the request is not compared to request templates on any subsequent directives.

The format of the directive is:

```
Fail  request-template [Server-IP-address or hostname]
```

request-template
A template for requests that you want your server to reject. If a request matches the template, the server sends the requester an error message.

You can use an asterisk as a wildcard in the template. The tilde character just after a slash (/) has to be explicitly matched; a wildcard cannot be used to match it.

Server-IP-address **or** *hostname*
If you are using multiple IP addresses or virtual hosts, use this parameter to

specify an IP address or a host name. (For more information on using multiple IP addresses or virtual hosts, see Chapter 17, "Running your server with multiple IP addresses or virtual hosts" on page 309.) The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

You can specify an IP address (for example, 204.146.167.72) or you can specify a host name (for example, hostA.bcd.com).

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

A wildcard character cannot be specified for a server's IP address.

Examples

```
Fail /usr/local/private/*
```

In the above example, the server rejects any requests beginning with /usr/local/private/.

```
Fail    /customerB/*    204.146.167.72
Fail    /customerA/*    9.83.100.45
```

The above example uses the optional IP address parameter. The server rejects any requests beginning with /customerB/ if the request comes in on a network connection with IP address 204.146.167.72. The server rejects any requests beginning with /customerA/ if the request comes in on a network connection with an IP address of 9.83.100.45.

```
Fail    /customerB/*    hostA.bcd.com
Fail    /customerA/*    hostB.bcd.com
```

The above example uses the optional hostname parameter. The server rejects any requests beginning with /customerB/ if the request comes in for hostA.bcd.com. The server rejects any requests beginning with /customerA/ if the request comes in for hostB.bcd.com.

Initial configuration file setting

None.

Map - Change matching requests to a new result string

Use this directive to specify a template for requests you want to change to a new request string. After your server changes the request, it takes the new request string and compares it to the request templates on subsequent directives.

The format of the directive is:

```
Map   request-template new-request [Server-IP-address or hostname]
```

request-template
A template for requests that you want your server to change and then continue comparing the new request string to other templates.

You can use an asterisk as a wildcard in the template. The tilde character just after a slash (/) has to be explicitly matched; a wildcard cannot be used to match it.

new-request
The new request string you want your server to continue to compare to the request templates on subsequent directives. *new-request* may contain a wildcard

if the *request-template* has one. The part of the request that matches the *request-template* wildcard is inserted in place of the wildcard in *new-request*.

Server-IP-address **or** *hostname*
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. (For more information on using multiple IP addresses or virtual hosts, see Chapter 17, "Running your server with multiple IP addresses or virtual hosts" on page 309.) The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

You can specify an IP address (for example, 204.146.167.72) or you can specify a host name (for example, hostA.raleigh.ibm.com).

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

A wildcard character cannot be specified for a server's IP address.

Examples

```
Map   /stuff/*   /good/stuff/*
```

In the above example, your server would take any requests starting with /stuff/ and change the /stuff/ portion of the request to /good/stuff/. Anything that followed /stuff/ on the original request would also be included in the new request string. So /stuff/whatsup/ would change to /good/stuff/whatsup/. Your server would take the new request string and continue to compare it to request templates on subsequent directives.

```
Map   /stuff/*   /customerA/good/stuff/*   204.146.167.72
Map   /stuff/*   /customerB/good/stuff/*   9.83.104.45
```

The above examples use the optional IP address parameter. If your server receives requests that begin with /stuff/, it changes the request to a different request string based on the IP address of the network connection the request comes in on. For requests coming in on 204.146.167.72, the server changes the /stuff/ portion of the request to /customerA/good/stuff/. For requests coming in on any connection with an address of 9.83.100.45, the server changes the /stuff/ portion of the request to /customerB/good/stuff/.

```
Map   /stuff/*   /customerA/good/stuff/*   hostA.bcd.com
Map   /stuff/*   /customerB/good/stuff/*   hostB.bcd.com
```

The above examples use the optional host name parameter. If your server receives requests that begin with /stuff/, it changes the request to a different request string based on the host name in the URL. For requests coming in for hostA, the server changes the /stuff/ portion of the request to /customerA/good/stuff/. For requests coming in for hostB, the server changes the /stuff/ portion of the request to /customerB/good/stuff/.

Initial configuration file setting

None.

Pass - Accept matching requests

Use this directive to specify a template for requests you want to accept and respond to with a document from your server. Once a request matches a template on a Pass directive, the request is not compared to request templates on any subsequent directives.

Resource mapping directives

The format of the directive is:

```
Pass  request-template [file-path [Server-IP-address or hostname]]
```

request-template
A template for requests that you want your server to accept and respond to with a document from your server.

You can use an asterisk as a wildcard in the template. The tilde character just after a slash (/) has to be explicitly matched; a wildcard cannot be used to match it.

file-path
The path to the file that contains the document you want the server to return. *file-path* may contain a wildcard if the *request-template* has one. The part of the request that matches the *request-template* wildcard is inserted in place of the wildcard in *file-path*.

This parameter is optional. If you do not specify a path, the request itself is used as the path.

Server-IP-address **or** *hostname*
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. (For more information on using multiple IP addresses or virtual hosts, see Chapter 17, "Running your server with multiple IP addresses or virtual hosts" on page 309.) The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

You can specify an IP address (for example, 204.146.167.72) or you can specify a host name (for example, hostA.raleigh.ibm.com).

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

A wildcard character cannot be specified for a server's IP address.

Examples

```
Pass   /updates/parts/*   /usr/lpp/internet/server_root/pub/*
```

In the above, your server would respond to a request starting with /updates/parts/ with a document from /usr/lpp/internet/server_root/pub/. Anything that followed /updates/parts/ would also be used to identify the document. So your server would respond to the request /updates/parts/printers/ribbon.html with the document in file /usr/lpp/internet/server_root/printers/ribbon.html.

```
Pass   /gooddoc/*
```

In the above example, your server would respond to a request starting with /gooddoc/ with a document from /gooddoc. So your server would respond to the request /gooddoc/volume1/issue2/newsletter4.html with the document in file /gooddoc/volume1/issue2/newsletter4.html.

```
Pass   /parts/*   /customerA/catalog/*   204.146.167.72
Pass   /parts/*   /customerB/catalog/*   9.83.100.45
```

The above examples use the optional IP address parameter. If your server receives requests that begin with /parts/, it returns a file from a different directory based on the IP address of the network connection the request comes in on. For requests

coming in on 204.146.167.72 the server returns a file from /customerA/catalog/. For requests coming in on any connection with an address of 9.83.100.45, the server returns a file from /customerB/catalog/.

```
Pass   /parts/*   /customerA/catalog/*   hostA.bcd.com
Pass   /parts/*   /customerB/catalog/*   hostB.bcd.com
```

The above examples use the optional host name parameter. If your server receives requests that begin with /parts/, it returns a file from a different directory based on the host name in the URL. For requests coming in for hostA, the server returns a file from /customerA/catalog/. For requests coming in for hostB, the server returns a file from /customerB/catalog/.

Initial configuration file setting

```
Pass   /icons/*      /usr/lpp/internet/server_root/icons/*
Pass   /Admin/*      /usr/lpp/internet/server_root/Admin/*.html
Pass   /Admin/*      /usr/lpp/internet/server_root/Admin/*.gif
Pass   /Docs/*       /usr/lpp/internet/server_root/Docs/*
Pass   /img-bin/*    /usr/lpp/internet/server_root/img-bin/*
Pass   /reports/*    /usr/lpp/internet/server_root/pubs/reports/*
Pass   /*            /usr/lpp/internet/server_root/pub/*
```

The path used for `Pass /*` is your document root directory.`Pass /*` matches all local server requests that do not match any other Pass directives.

Note: Only protect and pass rules for proxy should follow `Pass /*`.

Redirect - Send matching requests to another server

Use this directive to specify a template for requests you want to accept and send to another server. Once a request matches a template on a Redirect directive, the request is not compared to templates on any other directives in your configuration file.

The format of the directive is:

`Redirect` *`request-template URL`* [*`Server-IP-address`* or *`hostname`*]

request-template
: A template for requests that you want your server to send to another server.

You can use an asterisk as a wildcard in the template. The tilde character just after a slash (/) has to be explicitly matched; a wildcard cannot be used to match it.

URL
: The URL request you want your server to send to another server. The response to this request goes to the original requester without any indication that it did not come from your server.

URL must contain a protocol specification and the name of the server to send the request to. It can also contain a path or file name. If *request-template* uses a wildcard, the path or file name on *URL* can also use a wildcard. The part of the original request that matches the wildcard on *request-template* is inserted in place of the wildcard on *URL*.

Server-IP-address **or** *hostname*
: If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. (For more information on using multiple IP addresses or virtual hosts, see Chapter 17, “Running your server with multiple IP addresses or virtual hosts” on page 309.) The server uses the

directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

You can specify an IP address (for example, 204.146.167.72) or you can specify a host name (for example, hostA.bcd.com).

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

A wildcard character cannot be specified for a server's IP address.

Example

```
Redirect  /chief/stuff/*   http://www.other.org/wahoo/*
```

In this example, your server sends any requests beginning with /chief/stuff/ to the wahoo directory of the www.other.org server.

```
Redirect  /stuff/*   http://www.chief.org/wahoo/*   204.146.167.72
Redirect  /stuff/*   http://www.dawg.com/pound/*    9.83.100.45
```

The above examples use the optional IP address parameter. If your server receives requests that begin with /stuff/, it redirects the request to different servers based on the IP address of the network connection the request comes in on. For requests coming in on 204.146.167.72, the server sends the request to the wahoo directory of the www.chief.org server. For requests coming in on any connection with an address of 9.83.100.45, the server sends the request to the pound directory of the www.dawg.com server.

```
Redirect  /stuff/*   http://www.chief.org/wahoo/*   hostA.bcd.com
Redirect  /stuff/*   http://www.dawg.com/pound/*    hostB.bcd.com
```

The above examples use the optional IP address parameter. If your server receives requests that begin with /stuff/, it redirects the request to different servers based on the host name in the URL. For requests coming in for hostA, the server sends the request to the wahoo directory of the www.chief.org server. For requests coming in for hostB, the server sends the request to the pound directory of the www.dawg.com server.

Initial configuration file setting

None.

InheritEnv - Specify which environment variables are inherited by CGI programs

Use this directive to specify which environment variables you want your CGI programs to inherit (other than the CGI environment variables that are specific to CGI processing).

If you do not include an InheritEnv directive, all environment variables are inherited by CGI programs. If you include any InheritEnv directive, only those environment variables specified on InheritEnv directives will be inherited along with the CGI-specific environment variables. The directive allows you to optionally initialize the value of the variables that are inherited.

For a list of the CGI-specific environment variables, see Chapter 18, "Writing Common Gateway Interface programs" on page 315.

Example

```
InheritEnv PATH
InheritEnv LANG=ENUS
```

In this example, only the PATH and LANG environment variables will be inherited by CGI programs and the LANG environment variable will be initialized with the value of ENUS.

Initial configuration file setting

None.

Program default setting

None.

DisInheritEnv - Specify which environment variables are disinherited by CGI programs

Use this directive to specify which environment variables you do not want your CGI programs to inherit (other than the CGI environment variables that are specific to CGI processing).

By default, all environment variables are inherited by CGI programs. You can exclude individual environment variables from being inherited with the DisInheritEnv directive.

For a list of the CGI-specific environment variables, see Chapter 18, "Writing Common Gateway Interface programs" on page 315.

Example

```
DisInheritEnv PATH
DisInheritEnv LANG
```

In this example, all environment variables except PATH and LANG will be inherited by CGI programs.

Initial configuration file setting

None.

Program default setting

None.

Security - Set up secure connections for the server

Use the directives in this section to set up a secure Web server. For examples, see "Examples: Setting up secure connections" on page 69.

> **Important:**
> If you change any of the Security directives in this section, you must stop the Web server and start it again for the change to take effect.

KeyFile - Specify the name of a key database or SAF key ring to be used for SSL connections

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to register a key database or SAF key ring with the Web server.

In this book, the terms `key database` and `key ring` are similar but not the same. The term `key database` refers to a key database file (*.kdb) you create using the z/OS System SSL gskkyman utility. This key database file contains public keys, private keys, trusted CAs, and certificates. In RACF or other SAF security products, the term `key ring` does not refer to a key database file. A key ring contains public keys, private keys, and certificates, and is stored in a RACF or other SAF database.

You must specify the SAF option when you use key rings created using a SAF security product such as RACF. For planning considerations, see "Key rings defined in a SAF security product" on page 11.

If you have multiple KeyFile directives in the configuration file, the Web server uses the key database or key file on the last KeyFile directive as the default.

Example

In the following example, the key ring `safring7` is defined using the RACF RACDCERT ADDRING command and is stored in a RACF database:

```
KeyFile key.kdb
KeyFile safring7 SAF
```

This example specifies that the default key ring for the server is `safring7`.

For additional examples, see "Examples: Setting up secure connections" on page 69.

Initial configuration file setting

```
KeyFile key.kdb
```

Program default setting

```
None
```

NormalMode - Turn port on or off for HTTP connections

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to turn the port defined by the Port directive on or off.

Set NormalMode on for an HTTP connection. If you also want an SSL connection, set SSLMode on.

Note: If both NormalMode and SSLMode are turned off, the server will start in normal mode, and you will not have a secure network connection.

Initial configuration file setting

```
NormalMode on
```

Program default setting

```
NormalMode on
```

SSLCipherSpec - Specify the levels of encryption to use for SSL connections

Important:
If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify which cipher specifications the Web server will support and use for SSL connections. For more information on encryption support, see "Encryption support for the HTTP Server" on page 64.

Each encoded cipher specification is tested in the order specified for compatibility with the requester. If the requester supports an encryption level specified for the Web server, an SSL connection can be established. If not, the connection is refused.

If you do not have any Version 2 cipher specifications coded on the SSLCipherSpec directive, then SSL Version 2 is disabled. If you do not have any Version 3 cipher specifications coded on the SSLCipherSpec directive, then SSL Version 3 is disabled. If you do not have any SSLCipherSpec directives coded in the configuration file, SSL does not initialize. You must code at least one SSLCipherSpec directive for SSL to initialize.

The format of this directive is:

```
SSLCipherSpec cipher_specification
```

cipher_specification

A 2-digit code representing one of the cipher specifications that will be supported. Multiple SSLCipherSpec directives can be specified.

Valid codes for *cipher_specification* encryption:

The SSLCipherSpecs are listed in the order that is generally accepted by the industry as the order from strongest to weakest.

North American edition (US and Canada):

SSL V2:

27 Triple DES (192/168 bit)
21 RC4 (128 bit)
23 RC2 (128 bit)
26 DES (64/56 bit)
22 RC4 (40 bit)
24 RC2 (40 bit)

SSL V3:

3A Triple DES SHA (192/168 bit)
35 RC4 SHA (128 bit)
34 RC4 MD5 (128 bit)
39 DES SHA (64/56 bit)
33 RC4 MD5 (40 bit)
36 RC2 MD5 (40 bit)
32 NULL SHA
31 NULL MD5
30 NULL NULL

Note: Cipher specifications **32**, **31**, and **30** are standard SSL cipher specifications. However, they do not cause the data to be encrypted, and therefore do not provide data security. We only recommend them for debugging purposes.

International export edition:

SSL V2:
22 RC4 (40 bit)
24 RC2 (40 bit)

SSL V3:
33 RC4 MD5 (40 bit)
36 RC2 MD5 (40 bit)
32 NULL SHA
31 NULL MD5
30 NULL NULL

Note: Cipher specifications **32**, **31**, and **30** are standard SSL cipher specifications. However, they do not cause the data to be encrypted, and therefore do not provide data security. We only recommend them for debugging purposes.

Example

```
SSLCipherSpec 3A
SSLCipherSpec 35
SSLCipherSpec 27
SSLCipherSpec 26
```

You can specify the order of preference within a version of the cipher specification. The version of the cipher specification that is used is negotiated between the browser and the SSL security code. This example specifies the order of preference for the

- Version 3 cipher specification.
- Version 2 cipher specification.

Code	Cipher specification
3A	SSL V3-Triple DES SHA
35	SSL V3-RC4 SHA
27	SSL V2-Triple DES
26	SSL V2-DES

Any other cipher specifications requested by clients will result in an SSL handshake failure, and an error message will be logged in the error file. If the required specification is not included in the directive, an error message is logged in the error file.

Initial configuration file setting

```
All available cipher specifications are enabled.
```

Program default setting

```
All available cipher specifications are enabled.
```

SSLClientAuth - Select the type of SSL client authentication

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to select the type of SSL client authentication you want to use. When you use client authentication, the server requests a certificate from each client that makes an **https** request. Using client authentication increases network traffic.

The SSLClientAuth directive can be specified in any of the following formats:

OFF The secure server does not request certificates from clients.

LOCAL

The secure server requests certificates from clients. The server validates clients by checking for trusted CA certificates in the local key database. A trusted CA certificate is a certificate signed by a certificate authority who is designated as a trusted CA on your server.

PASSTHRU

The secure server requests certificates from clients but does not do any validity checking. This option should be used only be used if alternate validation methods are available, for example, checking performed by a GWAPI or CGI program.

For more information on GWAPI and CGI programs, see:
- Chapter 18, "Writing Common Gateway Interface programs" on page 315
- Chapter 19, "Writing GWAPI programs" on page 345

STRONG

The secure server retrieves trusted CA certificates from the X.500 directory server specified on the SSLX500Host directive.

If the certificate authority software used for the client certificate is provided by the IBM Registry and certificate revocation lists (CRLs) are supported in the certificate, the secure server retrieves them. For information on the IBM Registry, go to URL http://www.ibm.com/software/commerce/registry/.

Note: If you specify **STRONG** authentication, you must specify the host name or IP address of the X.500 directory server using the SSLX500Host directive. Other directives that control the use of strong authentication are SSLX500Port, SSLX500UserID, SSLX500Password, SSLV2Timeout, and SSLV3Timeout.

Examples

For client authentication examples, see "Examples: Setting up secure connections" on page 69.

Initial configuration file setting

```
SSLClientAuth off
```

Program default setting

```
SSLClientAuth off
```

SSLMode - Turn SSL on or off

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to turn the port defined by the SSLPort directive on or off.

Set SSLMode on for an SSL connection. If you also want an HTTP connection, set NormalMode on.

Note: If both NormalMode and SSLMode are turned off, the server starts in normal mode with an insecure connection.

Initial configuration file setting

```
SSLMode on
```

Program default setting

```
SSLMode on
```

SSLPort - Set port for SSL security

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to set the port for SSL security. The server will use this port only for HTTPS requests. (Requests for HTTP will still come on the port that you set with the Port directive.)

If you want to use a port other than 443, specify a port above 1024.

Initial configuration file setting

```
SSLPort 443
```

Program default setting

```
SSLPort 443
```

SSLServerCert - Associate a key database with an IP address

Important:
If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

This directive can be used whether your server is configured with a single IP address or multiple IP addresses. If your server is configured with multiple IP addresses, use this directive to assign a key database label to each IP address. For a description of this option, see "SSL support for multiple IP addresses" on page 62.

The format of this directive is:

```
SSLServerCert keyname  IP-address
```

keyname
The name of the entry in the key database file that contains the keys and certificate for the specified host name-IPaddress combination. The value for the *keyname* parameter can contain blanks and must include a delimiter. Any delimiter can be used. The server assumes the first and last characters are the delimiters and must be the same character. All entries for the *keyname* parameter must be in the same key database as specified by the KeyFile directive.

IP-address
The IP address assigned to a host name configured for the server.

Examples

In the following examples, `My Server's Certificate` is the certificate label in the key database file.

```
SSLServerCert  "My Server's Certificate"    19.123.55.12
SSLServerCert  /My Server's Certificate/    19.123.55.12
SSLServerCert  zMy Server's Certificatez    19.123.55.12
```

The following examples will result in errors during server initialization:

```
SSLServerCert  My Server's Certificate    19.123.55.12
SSLServerCert  'My Server's Certificate'  19.123.55.12
SSLServerCert  19.123.55.12
```

Initial configuration file setting

```
None.  An example is included in the server configuration
       file for all of its IP addresses.
```

Program default setting

```
None.
```

Hints and Tips

- SSLServerCert can be placed anywhere in the configuration file and can appear multiple times.
- If more than one key database label is specified for the same IP address, the last value is the one that is used.
- If SSLServerCert is not specified and SSLMode is on, the server uses the key and certificate marked as the default in the key database file.

- If SSLServerCert is specified for some IP addresses (and not for others), the default key and certificate are used for those not specified on the SSLServerCert keyword.

SSLV2Timeout — Set SSL V2 session timeout value

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the length of time that a browser can reuse an SSL V2 session ID without renegotiating encryption keys with the server.

Note: You can specify a value in the range 1-100 seconds. If you specify a value outside this range, SSL initialization will fail with error message IMW6310E and the Web server will run in normal mode.

Initial configuration file setting

```
SSLV2Timeout 100 seconds
```

Program default setting

```
SSLV2Timeout 100 seconds
```

SSLV3Timeout — Set SSL V3 session timeout value

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the length of time that a browser can reuse an SSL V3 session ID without renegotiating encryption keys with the server.

Note: You can specify a value in the range 1-86400 seconds. If you specify a value outside this range, SSL initialization will fail with error message IMW6310E and the Web server will run in normal mode.

Initial configuration file setting

```
SSLV3Timeout 1000 seconds
```

Program default setting

```
SSLV3Timeout 1000 seconds
```

SSLX500CARoots — Specify location of trusted CA certificates

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify where the secure server will search for trusted CA certificates to validate client certificates.

The SSLX500CARoots directive can be specified in any of the following formats:

LOCAL_ONLY

The secure server uses trusted CA certificates stored in the server's local key database.

LOCAL_AND_X500

The secure server uses trusted CA certificates stored in the server's local key database and in an X.500 directory server.

The server first checks the local key database. If a certificate is not found, the server searches the X.500 directory server. Certificates are retrieved from an X.500 directory server using the Lightweight Directory Access Protocol (LDAP). For more information on LDAP support and options, see Chapter 15, "Retrieving Lightweight Directory Access Protocol information" on page 297.

Initial configuration file setting

```
SSLX500CARoots local_only
```

Program default setting

```
SSLX500CARoots local_only
```

SSLX500Host — Specify host name or IP address of the X.500 directory server

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify either the fully qualified host name or IP address of the X.500 directory server. This directive is required if you specify **STRONG** on the SSLClientAuth directive.

Examples:

```
SSLX500Host    www.my_x500_host.com
SSLX500Host    9.97.123.176
```

Initial configuration file setting

None

Program default setting

None

SSLX500Port — Specify X.500 directory server port number

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the port number used for communicating with the X.500 directory server. Specify a value in the range 1-64000.

Example:

```
SSLX500Port  22343
```

Initial configuration file setting

None

Program default setting

None

SSLX500UserID — Specify user ID for the LDAP connection to the X.500 directory server

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the distinguished name of the user ID for the Lightweight Directory Access Protocol (LDAP) connection to the X.500 directory server. If you do not specify a distinguished name, **anonymous** will be used.

Initial configuration file setting

None

Program default setting

None

SSLX500Password — Specify user ID password for the LDAP connection to the X.500 directory server

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use this directive to specify the password of the distinguished name of the user ID for the Lightweight Directory Access Protocol (LDAP) connection to the X.500 directory server. This password is valid only if SSLX500UserID has been specified.

You can use the following characters and digits when specifying the password: A-Z, a-z, 0-9.

Initial configuration file setting

`None`

Program default setting

`None`

System Management - Define system management settings

Use the directives described in this section to define and tailor settings used to monitor and manage the performance, throughput, and activity of your servers.

ApplEnv - Specify application environment for workload management

Use this directive to specify the application environments and workload management transaction classes for processing URL requests matching a given template. Each application environment is assigned its own transaction queue. This directive tells the server how to associate requests with a queue. The format of the directive is:

```
ApplEnv uri_template AEName [WLM_transaction_class [ip_addr_template]]
```

ApplEnv
Specifies a unique name for each queue

url_template
A template for requests that you want your server to accept and respond to with a document from your server. You can use an asterisk as a wildcard in the template. The tilde character just after a slash (/~) has to be explicitly matched; a wildcard cannot be used to match it.

AEName
Represents the application environment name comprised of any alphanumeric name, up to thirty-two characters. Use an alphanumeric application environment name to forward all work to an explicitly defined queue server. Then you take full advantage of Workload Management (WLM) availability features. Define any application environment names that are referenced by using this directive, under Application Environments in the WLM application panels. For more information about WLM application panels, see "Workload Management Enablement for the Web server" on page 228 or *z/OS MVS Planning: Workload Management*. To access this book on the Web, go to the z/OS

Book Server Web site at URL:
`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

Note: You can represent the application environment name as a wildcard (*). Although supported, the wild card is not recommended. If you use the wildcard, the following happens. All URL requests that do not map to an application environment name on other ApplEnv directives go to the queue manager.

WLM_transaction_class
Optionally, you can specify the WLM transaction class. The transaction class is an input to WLM classification. Your WLM classification rules for IWEB can use the transaction class qualifier to map the transaction class to a service class. The transaction class can be any alphanumeric name up to eight characters or a single wildcard (*). If a wildcard is specified, the transaction class is not provided to WLM classification. The HTTP Server provides the following input for classification:

```
TRANSACTION_CLASS: WLM_transaction_class (if not *)
TRANSACTION_NAME:      HTTP Request Line Method (for example, GET, POST)
USERID:            Server's LOGIN user ID
SUBSYSTEM_PARM:        Subsystem PARM buffer:
                          0..7  Subsystem name (-SN xxx)
                             8  Blank
                         9..23  Source IPAddr nnn.nnn.nnn.nnn
                            24  Blank
                        25..39  Target IPAddr nnn.nnn.nnn.nnn
                            40  Blank
                        41..46  Target Port nnnnnn
                                  47  X'00'
```

Any transaction classes that are referenced using this directive MUST also be defined in the Classification Rules for IWEB using the TN (transaction name) qualifier in the WLM application panels. For more information about WLM application panels, see "Workload Management Enablement for the Web server" on page 228 or the *z/OS MVS Planning: Workload Management* book.

ip_addr_template
If you are using multiple IP addresses, use this parameter to specify which IP address this directive applies to.

Example

```
APPLENV /Admin/*.html  WEBHTML  ADMIN
```

Initial configuration file setting

None

ApplEnvConfig - Specify a set of directives to tailor the application environment

Use this directive to specify a set of directives to tailor application environments to support various workloads and conserve resources. Specify a subset of directives so your application environments are each configured appropriately. Each application environment can be specifically configured to handle the requests that are processed for that Application Environment.

```
   ApplEnvConfig AEName|%%QS%%|%%QM%% {
directive
directive
directive
}
```

ApplEnvConfig
Specifies a set of directives to tailor a particular application environment

AEName
The application environment name can be any alphanumeric name up to thirty-two characters that is referenced by using this directive and MUST also be defined under application environments in the WLM application panels. In other words, this must be one of the AENames specified on the ApplEnv directive. Refer to "ApplEnv - Specify application environment for workload management" on page 555.

For more information about WLM application panels, see "Workload Management Enablement for the Web server" on page 228 of the *z/OS MVS Planning: Workload Management* book. You can replace *AEName* with either of the following:

%%QS%%
This enables directives specified within the ApplEnvConfig to affect all queue servers. Use this statement to distinguish between the queue manager and the queue servers.

%%QM%%
This enables directives specified within the ApplEnvConfig set to affect the queue manager only. Use %%QM%% to specify directives exclusively for the queue manager, because the queue manager does not have an application ennvironment name.

directive
Any of these:

- PluginInclude
- PluginExclude
- PluginDefault Include/Exclude
- CacheLocalFile
- ServerInit
- MaxActiveThreads
- ApplEnvMin
- ApplEnvMax

When the directives listed above are used within the ApplEnvConfig directive, they apply to a particular application environment. When you specify them outside an ApplEnvConfig statement, they apply globally. When the server runs standalone, all statements within the ApplEnvConfig set are ignored. Global directives are not ignored when running standalone.

The three Plugin directives are described later in this section. Read about ApplEnvMin, ApplEnvMax, CachelLocalFile, ServerInit, and MaxActiveThreads directives in previous pages in this section. Using the CachelLocalFile, ServerInit or MaxActiveThreads directives within an ApplEnvConfig statement allows you to tailor what files get cached, which ServerInit Plugins are used (and what parameters are passed), and the number of active threads, respectively.

Read more about how to implement these directives in "Using configuration directives to tailor your application environments" on page 561.

Example

```
ApplEnvConfig WEBHTML {
ApplEnvMin 2
ApplEnvMax 4
CacheLocalFile /www/powerco/*.html
MaxActiveThreads 50
PluginInclude  /ics/api/bin/icsexit05.so
ServerInit /usr/lpp/internet/bin/mvsds.so:mvsdsInit /u/webserv/config/mymvsds.conf
}
```

Initial configuration file setting

None.

Program default setting

None.

ApplEnvMax - Specify maximum number of Queue Servers that WLM is to maintain for a specific application environment

You can specify the maximum number of Queue Servers that WLM is to maintain for a specific ApplEnv. The ApplEnvMax directive can be used within the ApplEnvConfig directive. A value of 0 means that no maximum is to be set. The directive format is:

```
ApplEnvMax number
```

number is the maximum number of queue servers that WLM maintains for a specific ApplEnv.

Example

```
ApplEnvMax 3
```

Note: If you want to have the necessary queue servers active for any incoming requests, do the following. Make sure that the ApplEnvMax value is at least as large as the maximum number of service classes that can map to the ApplEnv. If it is smaller, then WLM could be prevented from creating a queue server to do a request for a different service class. This is because WLM will only run work of a single service class in a queue server at one time.

Initial configuration file setting

None.

Program default setting

None.

ApplEnvMin - Specify minimum number of Queue Servers that WLM is to maintain for a specific application environment

You can specify the minimum number of Queue Servers that WLM is to maintain for a specific ApplEnv. The ApplEnvMin directive can be used within the ApplEnvConfig directive. WLM will not manage the number of Queue Servers for a particular ApplEnv until a request for that ApplEnv is queued. The ApplEnvPrestart directive can be used to queue a dummy request at initialization which will then trigger WLM to start the minimum number. A value of 0 means that no minimum is to be set. The directive format is:

```
ApplEnvMin number
```

number is the minimum number of queue servers that WLM maintains for a specific ApplEnv.

Example

```
ApplEnvMin 2
```

Note: If you want to have the necessary queue servers active for any incoming requests, do the following. Make sure that the ApplEnvMin value is at least as large as the number of service classes that can map to the ApplEnv. This is because WLM will only run work of a single service class in a queue server at one time.

Initial configuration file setting

None.

Program default setting

None.

ApplEnvPrestart - Specify an application environment to start at initialization

Specify ApplEnvPrestart to start an ApplEnv at initialization time. If this directive is specified, a dummy request is queued to the specified ApplEnv. This dummy request will cause WLM to start a Queue Server for that ApplEnv. ApplEnvPrestart can be used with or without ApplEnvMin. The directive format is:

```
ApplEnvPrestart AEName [WLM_transaction_name] [WLM_transaction_class]
```

AEName
The application environment name can be any alphanumeric name up to thirty-two characters that is referenced by using this directive and MUST also be defined under application environments in the WLM application panels. In other words, this must be one of the AENames specified on the ApplEnv directive. Refer to "ApplEnv - Specify application environment for workload management" on page 555.

For more information about WLM application panels, see "Workload Management Enablement for the Web server" on page 228 of the *z/OS MVS Planning: Workload Management* book. You can replace *AEName* with either of the following:

WLM_transaction_name
The transaction name is set to the httpd METHOD of the request. For example, it is set to GET | POST | HEAD, etc. The transaction name is optional and will default to GET if not specified. This name can associate work with WLM policies and specifies how work should perform.

WLM_transaction_class
The transaction class must be known to WLM. The transaction class is optional. This class can associate work with WLM policies. The class specifies how the work should perform.

Example

```
ApplEnvPrestart WEBHTML GET FAST
```

Note: WLM will only run work of a single service class in a queue server at one time. If there could be different service classes within the same ApplEnv,

then make sure that the queue servers that get started at initialization are bound to service classes of the workload. If the WLM_transaction_name and the WLM_transaction_class are used to classify work requests to different service classes, then do the following:

- Specify the WLM_transaction_name and the WLM_transaction_class on ApplEnvPrestart.
- Do an ApplEnvPrestart for each service class within an ApplEnv.

Initial configuration file setting

`None.`

Program default setting

`None.`

PluginDefault - Specify default Plugin action

Use this directive to include or exclude Plugins by default. User Plugins specified on GWAPI directives will not be invoked unless loaded during initialization. The system default is to load all referenced user Plugins. The directive format is:

```
PluginDefault Include|Exclude
```

Include
Sets the default action to include user Plugins that have not been specifically referenced with a PluginInclude or a PluginExclude directive.

Exclude
Sets the default action to exclude user Plugins that have not been specifically referenced with a PluginInclude or PluginExclude directive.

Example

```
PluginDefault Exclude
```

Initial configuration file setting

`None.`

Program default setting

```
PluginDefault Include
```

PluginExclude - Specify Plugin not to load during initialization

Use this directive to indicate that the specified user Plugin should **not** be loaded during initialization. Plugins specified on GWAPI directives will not be invoked effect unless loaded during initialization. The system default is to load all referenced user Plugins. The directive format is:

```
PluginExclude /path/file
```

/path/file
The fully qualified file name of your compiled DLL, including the extension.

Example

```
PluginExclude /ics/api/bin/icsexit05.so
```

Initial configuration file setting

`None.`

Program default setting

`None.`

PluginInclude - Specify Plugin to be loaded during initialization

Use this directive to indicate that the specified user Plugin should be loaded during initialization. Plugins specified on GWAPI directives will not be invoked unless loaded during initialization. The system default is to load all referenced user Plugins. The format of the directive is:

```
PluginInclude /path/file
```

/path/file
The fully qualified file name of your compiled DLL, including the extension.

Example

```
PluginInclude /ics/api/bin/icsexit05.so
```

Initial configuration file setting

```
None.
```

Program default setting

```
None.
```

Using configuration directives to tailor your application environments

PluginInclude, PluginExclude, and PluginDefault directives, in conjunction with the ApplEnvConfig directive, let you tailor which user DLLs are loaded in each of the Web server environments running WLM mode. Save storage by avoiding loading user DLLs that will not be invoked. Also, disable user Plugins by not loading the DLLs that contain them.

To learn more about these directives, refer to "ApplEnvConfig - Specify a set of directives to tailor the application environment" on page 556. To tailor your application environments for maximum performance, reevaluate your existing GWAPI Plugins and related directives.

Implicit actions of PluginInclude and PluginExclude directives: Be aware that PluginInclude and PluginExclude directives can generate implicit actions beyond what you may intend. Consider the following example:

```
ApplEnvConfig  WEBHTML {
PluginInclude  /path/plugin.so
}
```

Besides causing DLL /path/plugin.so to be loaded when application environment WEBHTML is being configured, this specification also causes /path/plugin.so to **not** be loaded in the other web server application environments (queue servers) or the queue manager, assuming there are no other PluginInclude directives for /path/plugin.so.

If a DLL is needed in only one application environment, only one PluginInclude is needed within an ApplEnvConfig directive. This ensures that the DLL is loaded only for that application environment, and not loaded unnecessarily in other places.

Directives explicitly coded take precedence over implicit actions. If conflicting implicit specifications occur, the DLL is not loaded.

Order of precedence of ApplEnvConfig directives: A definite precedence order is determined by the "match" of the directive. When configuring a queue server, a

directive within an ApplEnvConfig directive for the current application environment takes precedence over a match on a %%QS%%, which takes precedence over directives that are outside an ApplEnvConfig. Any explicit directives take precedence over implicit actions. The physical order matters only with multiple directives at the same level, and in some cases may even cause a syntax error.

The example below causes the application environment WEBHTML to have a MaxActiveThreads of value of 40, while all other queue servers would use 50. The queue manager would use 20. All queue servers would cache /other/*.html while WEBHTML would also cache the local files /path/*.html. The queue manager would not cache any. Both WEBHTML and the queue manager would load /mypath/mydll.so. All other queue servers would not load /mypath/mydll.so due to the generated implicit action. WEBHTML would also call the ServerInit function mvsdsInit passing "u/WEBSRV/config/mymvsds.conf".

```
ApplEnvConfig WEBHTML {
MaxActiveThreads 40
CacheLocalFile /path/*.html
PluginInclude /mypath/mydll.so
ServerInit /usr/lpp/internet/bin/mvsds.so:mvsdsInit u/WEBSRV/config/mymvsds.conf
}
ApplEnvConfig %%QS%% {
MaxActiveThreads 50
CacheLocalFile /other/*.html
}
ApplEnvConfig %%QM%% {
PluginInclude /mypath/mydll.so
}
MaxActiveThreads 20
```

The example below causes all user DLLs to **not** be loaded for the queue manager. This specification is appropriate if no requests are processed by the queue manager (for example, if all requests are routed to the application environments):

```
ApplEnvConfig %%QM%% {
PluginDefault Exclude
}
```

To tailor your application environment:

1. Evaluate your DLLs to see which are used in particular application environments. Identify DLLs to include and exclude for each different environment.

   Decide which directives to use within an ApplEnvConfig directive. If a DLL is needed only in one application environment, use the PluginInclude directive for that DLL within an AppEnvConfig statement for that application environment. This will also implicitly cause that DLL to not be loaded in other environments.

   PluginInclude Example

   ```
   ApplEnvConfig MYAE {
   PluginInclude /mypath/mydll.so
   }
   ```

   This example would cause the MYAE application environment to load /mypath/mydll.so and the other environments to not load it. Use the directives this way if you only want the DLL loaded in one application environment.

   Likewise, use PluginExclude if a DLL should be loaded everywhere except in a particular application environment.

PluginExclude Example

```
ApplEnvConfig NOTMYAE {
PluginExclude /mypath/mydll.so
}
```

This example would cause the NOTMYAE application environment to not load /mypath/mydll.so and the other environments to load it. Use the directives in this way if you want the DLL in all but the one application environment.

2. Use the Plugin Default Include/Exclude directive to specify default actions for an application environment.

   PluginDefault Example

   ```
   ApplEnvConfig %%QM%% {
   PluginDefault Exclude
   PluginInclude /mypath/mydll.so
   }
   ```

   This example would cause the Queue Manager to load /mypath/mydll.so and no other DLL's. Be careful when using PluginDefault Exclude, you could possibly exclude DLL's that you may want.

3. Use the CacheLocalFile directive within the ApplEnvConfig directive to cache additional files for particular application environments.

   CacheLocalFile Example

   ```
   CacheLocalFile /all/*.html
   ApplEnvConfig WEBAE1 {
   CacheLocalFile /AE1ONLY/*.html
   }
   ```

   This example would cause the WEBAE1 to cache the files in /AE1ONLY/*.html in addition to the files in /all/*.html.

4. Use the ServerInit directive within the ApplEnvConfig directive to vary the parameters with which a user Plugin is invoked.

   ServerInit Example

   ```
   ApplEnvConfig  WEBAE1 {
   ServerInit /usr/lpp/internet/bin/mvsds.so:mvsdsInit /u/config/mvsds1.conf
       }
   ApplEnvConfig WEBAE2 {
   ServerInit /usr/lpp/internet/bin/mvsds.so:mvsdsInit /u/config/mvsds2.conf
       }
   ```

   This example shows how you can have different application environments invoke a ServerInit plugin with different parameters.

5. Use the MaxActiveThreads directive within ApplEnvConfig to set a maximum number of threads to have active for a particular application environment.

SNMP - Enable and disable SNMP support

Use this directive to enable or disable SNMP support. The form of the directive is:

`SNMP` *`setting`*

The *setting* can have a value of on or off.

on SNMP support is turned on.

off
SNMP support is turned off.

Example

```
SNMP on
```

Initial configuration file setting

```
SNMP off
```

Program default setting

```
SNMP off
```

SNMPCommunityName - Provide a security password for SNMP

Use this directive to define the password used between the HTTP Server DPI subagent and the SNMP agent.

The form of the directive is:

```
SNMPCommunityName com_nam
```

Example

```
SNMPCommunityName public
```

Initial configuration file setting

```
SNMPCommunityName public
```

Program default setting

```
SNMPCommunityName public
```

WebMasterEmail - Create an e-mail address to receive SNMP problem reports

Use this directive to create an e-mail address to receive SNMP problem reports. The default mail address is `webmaster`.

The form of the directive is:

```
WebMasterEmail webmastermailaddress
```

Example

```
WebMasterEmail webmaster@computer.com
```

Initial configuration file setting

```
WebMasterEmail webmaster
```

Program default setting

```
WebMasterEmail webmaster
```

Timeouts - Close connections automatically

Use the directives described in this section to control the amount of time the server spends processing requests. If you are using persistent connections, see "PersistTimeout - Specify time to wait for the client to send another request" on page 576.

InputTimeout - Specify time allowed for the client to send a request

Use this directive to set the time allowed for a client to send a request after making a connection to the server. A client first connects to the server and then

sends a request. If the client does not send a request within the amount of time on this directive, the server drops the connection.

The format of this directive is:

```
InputTimeout  time
```

Specify *time* in any combination of hours, minutes (or mins), and seconds (or secs).

Examples

```
InputTimeout 1 hour
InputTimeout 3 mins 30 secs
InputTimeout 1 minute 30 seconds
```

Initial configuration file setting

```
InputTimeout 30 seconds
```

Program default setting

```
InputTimeout 30 seconds
```

OutputTimeout - Specify maximum time for sending output to the client

Use this directive to set the maximum time allowed for your server to send output to a client. The time limit applies to requests for local files and requests for which the server is acting as a proxy. The time limit does not apply to requests that start a local CGI program.

If the server does not send the complete response within the amount of time on this directive, the server drops the connection.

The format of this directive is:

```
OutputTimeout  time
```

Specify *time* in any combination of hours, minutes (or mins), and seconds (or secs).

Examples

```
OutputTimeout 1 hour
OutputTimeout 5 minutes 30 seconds
OutputTimeout 1 min 30 secs
```

Initial configuration file setting

```
OutputTimeout 2 minutes
```

Program default setting

```
OutputTimeout 2 minutes
```

ScriptTimeout - Specify time allowed for a program to complete

Use this directive to set the time allowed for a program started by the server to finish. When the time runs out, the server sends a message (**SIGTERM** signal) to the program. Five seconds later, the server sends a **KILL** signal.

The format of this directive is:

```
ScriptTimeout  time
```

Specify *time* in any combination of hours, minutes (or mins), and seconds (or secs).

Examples

```
ScriptTimeout 1 hour
ScriptTimeout 5 mins
ScriptTimeout 2 minutes 30 seconds
```

Initial configuration file setting

```
ScriptTimeout 2 minutes
```

Program default setting

```
ScriptTimeout 2 minutes
```

Tuning - Define performance and scalability settings

Use the directives described in this section to tune your Web server. For planning considerations and recommendations, see "Performance" on page 9.

For the most current tuning information, see the WebSphere Troubleshooter for z/OS tuning hints and tips at URL:

`http://www.ibm.com/software/webservers/httpservers/doc/v51/ttunc.htm`

To enable and customize caching using the Fast Response Cache Accelerator, see "EnableFRCA — Turn dynamic caching on or off" on page 567. For more information on the Cache Accelerator, see Chapter 10, "Customizing cache management with the Fast Response Cache Accelerator" on page 185.

CacheLocalFile - Specify files you want to load in memory at start up

Use this directive to specify the names of files you want to load into the server's memory each time you start the server. You can have multiple occurrences of this directive in the configuration file. Include a separate directive for each file you want to load into memory.

By keeping your most frequently requested files loaded in the server's memory, you can improve your server's response time for those files. For example, if you load your server's welcome page into memory at startup, the server can handle requests for the page much more quickly than if it had to read the file from a disk. Keep in mind that for each file you load into memory, you are making that amount of memory unavailable for other uses.

Before responding to a request for a file that is stored in memory, the server checks to see if the file has changed since the server was started. If the file has changed, the server responds to the request with the updated file and deletes the old version from its memory. To load the new file into memory, you need to restart the server.

Notes:

1. You can use an asterisk (*) as a wildcard character on the file names.
2. File name matching is not recursive. Only files in the specified directory will be cached. No files in subdirectories are affected.

Example

To cache a specific file

```
CacheLocalFile   /www/html/index.html
```

To cache all .html files in the powerco directory

```
CacheLocalFile   /www/powerco/*.html
```

Initial configuration file setting
None.

Program default setting
None.

CacheLocalMaxBytes - Specify maximum amount of memory to use for file caching

Use this directive to specify the maximum amount of memory you want to allow for file caching. You can specify the memory in kilobytes (K) or megabytes (M). You must still use the CacheLocalFile directive to indicate which files you want cached.

Note: CacheLocalMaxBytes can help limit your cache size when you are using the wildcard character to specify the files on the CacheLocalBytes directive.

Example
```
CacheLocalMaxBytes   5K
```

Initial configuration file setting
```
CacheLocalMaxBytes   2M
```

Program default setting
```
CacheLocalMaxBytes   2M
```

CacheLocalMaxFiles - Specify the maximum number of files for caching

Use this directive to specify the maximum number of files you want to be cached at one time. You must still use the CacheLocalFile directive to indicate which files you want cached.

Note: CacheLocalMaxFiles can help limit your cache size when you are using the wildcard character to specify the files on the CacheLocalFile directive.

Example
```
CacheLocalMaxFiles   150
```

Initial configuration file setting
```
CacheLocalMaxFiles  200
```

Program default setting
```
CacheLocalMaxFiles  200
```

EnableFRCA — Turn dynamic caching on or off

Use this directive to enable dynamic caching using the Fast Response Cache Accelerator. To customize dynamic caching, use the following directives:
- FRCACacheSize
- FRCACacheEntries
- FRCACacheOnly or FRCANoCaching
- FRCAMaxFileSize
- FRCAAccessLog
- FRCAStackName
- FRCAVirtualHost
- FRCAWLMParms

For more information on the Cache Accelerator, see Chapter 10, "Customizing cache management with the Fast Response Cache Accelerator" on page 185.

Example

```
EnableFRCA   On
```

Initial configuration file setting

```
EnableFRCA   Off
```

Program default setting

```
EnableFRCA   Off
```

Enclave — Create and join a Workload Management enclave for the current request

> **Important:**
> If you change this directive, you must stop the Web server and start it again for the change to take effect. For a complete list, see "Directives that require a stop and restart of the server" on page 425.

Use the Enclave directive to create and join a Workload Mangement (WLM) enclave for the current request. This directive tells the server how to associate the request with a transaction class based on the URI template or the host IP address. Use this directive in conjunction with the WLMSN directive. For information on the WLMSN directive, see "WLMSN- Enable the creation of request level Workload Management enclaves" on page 578.

The Enclave directive performs a transaction class-only classification. The CLASSIFY environment variable in the Go Webserver Application Programming Interface (GWAPI) interface provides a more extensive classification. You can use the Enclave directive concurrently with the CLASSIFY environment variable. For information on the CLASSIFY environment variable, see Appendix D, "Environment variables" on page 583.

If a request matches a URI template on an Enclave directive, the Web server opens the enclave under the Enclave directive classification. The Web server opens an enclave that uses the CLASSIFY environment variable only if the request does not match a URI template on an Enclave directive. The enclave remains open under the specified classification until request processing completes. Only one enclave at a time is open.

Make the last Enclave directive `Enclave /*`, as long as you do not use the Enclave directive in conjunction with the CLASSIFY environment variable. This guarantees that every request generates an enclave under the Enclave directive classification.

The format of the directive follows:

```
Enclave uri-template  WLM_transaction_class [IP-address]
```

uri_template
: A template for requests that you want your server to accept and respond to with a document from your server. You can use an asterisk as a wildcard in the template. The tilde character just after the slash (/) has to explicitly match. You cannot use a wildcard to match it.

WLM_transaction_class
The transaction class is the input to WLM classification. Your WLM classification rules for the IWEB subsystem type use the transaction class qualifier to map the transaction class to a service class. Use any alphanumeric name up to eight characters for the transaction class.

Insure that any transaction classes you define on the Enclave directive you also define in the WLM application panels. Define these classes in the Classification Rules for the IWEB subsystem type, by using the transaction name (TN) qualifier. For more information about WLM application panels, see "Workload Management Enablement for the Web server" on page 228, or *z/OS MVS Planning: Workload Management*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

IP-address
If you use multiple IP addresses, use this parameter to specify which IP address the Enclave directive applies to.

Note: This parameter only applies to IP addresses. You cannot use a host name on this parameter.

Example

```
Enclave /Admin/*.html ADMIN
```

Initial configuration file setting

```
None
```

Program default setting

```
None
```

FRCAAccessLog — Specify a log file path and name for dynamic caching requests

Use this directive if you want requests served by the Fast Response Cache Accelerator to be logged in a separate log file. If this directive is used, no other log entries will be written for requests served from the Cache Accelerator cache. If this directive is not used, requests that are served by the Cache Accelerator will be logged the same as other requests.

You may want to use this optional log if you are concerned about performance but still want access log information about requests served from the Cache Accelerator cache.

Note: The server will ignore this setting if the EnableFRCA directive is set off.

Example

```
FRCAAccessLog   /usr/lpp/internet/server_root/logs/frca-log
```

Initial configuration file setting

```
None
```

Program default setting

```
None
```

FRCACacheEntries — Specify the maximum number of files to be dynamically cached

Use this directive to specify the maximum number of individual files that will be cached by the Fast Response Cache Accelerator. Your z/OS Communications Server support determines the maximum number of files that can be cached.

Note: The server will ignore this setting if the EnableFRCA directive is set off.

Example

```
FRCACacheEntries   500
```

Initial configuration file setting

```
FRCACacheEntries   1000
```

Program default setting

```
FRCACacheEntries   1000
```

FRCACacheOnly — Specify URIs to be dynamically cached

Use this directive to specify a set of URIs that will be considered for caching by the Fast Response Cache Accelerator. URIs that are not in this list will never be cached.

Notes:

1. The server will ignore this setting if the EnableFRCA directive is set off.
2. Use either this directive or the FRCANoCaching directive.

Example

```
FRCACacheOnly *.gif
FRCACacheOnly requests/*.html
FRCACacheOnly faqs/*.html
```

Initial configuration file setting

```
None
```

Program default setting

```
None
```

FRCACacheSize — Specify size of the dynamic cache

Use this directive to specify the size of the Fast Response Cache Accelerator cache by entering the number of 4K (4096-byte) blocks of memory you want to allocate. The maximum size is limited by the amount of available memory in the z/OS Communications Server.

Note: The server will ignore this setting if the EnableFRCA directive is set off.

Example

```
FRCACacheSize   500
```

In this example, the amount of memory allocated for the Cache Accelerator cache is 2000K (500 x 4K).

Initial configuration file setting

```
FRCACacheSize   25000
```

Program default setting

```
FRCACacheSize   25000
```

FRCAMaxFileSize — Specify maximum file size for the dynamic cache

Use this directive to specify the maximum size of a single file that will be placed in the Fast Response Cache Accelerator cache. You can specify this value in bytes, kilobytes, or megabytes. There is no minimum size.

Note: When you specify the cache size, use the following guidelines:
- For bytes, enter an integer
- For megabytes, enter an integer followed by M
- For kilobytes, enter an integer followed by K

Your z/OS Communications Server support determines the maximum size of files that can be cached.

Note: The server will ignore this setting if the EnableFRCA directive is set off.

Example

```
FRCAMaxFileSize   2 M
```

Initial configuration file setting

```
FRCAMaxFileSize   1 M
```

Program default setting

```
FRCAMaxFileSize   1 M
```

FRCANoCaching — Exclude URIs from the dynamic cache

Use this directive to specify a set of URIs which must never be cached by the Fast Response Cache Accelerator. URIs that are not specified in this list will be considered for caching.

Notes:

1. The server will ignore this setting if the EnableFRCA directive is set off.
2. Use either this directive or the FRCACacheOnly directive.

Example

```
FRCANoCaching   /usergroup/*.html
FRCANoCaching   /Docs/*.pdf
```

Initial configuration file setting

```
None
```

Program default setting

```
None
```

FRCAStackName — Specify the TCP/IP stack that supports the dynamic cache

This directive is needed only if you are using the z/OS UNIX System Services common INET function with multiple TCP/IP stacks.

Specify the name of the z/OS UNIX physical file system that supports the TCP/IP stack used by the Fast Response Cache Accelerator. This name must be 8 characters or less, and it should match the name on the SubFileSysType statement in the z/OS UNIX BPXPRMxx parmlib member.

Note: The server will ignore this setting if the EnableFRCA directive is set off.

Example

```
FRCAStackName   TCP34
```

Initial configuration file setting

```
FRCAStackName   TCP34
```

Program default setting

```
None
```

FRCAVirtualHost — Indicate to the dynamic cache whether multiple virtual hosts or IP addresses are being used

Use this directive to indicate to the Fast Response Cache Accelerator cache whether the Web server is configured to use multiple virtual host names or IP addresses. The Cache Accelerator cache uses this setting to expand the cache search key structure to include the use of multiple host names or IP addresses.

Valid settings are:

AUTO (default)
The Web server determines automatically whether multiple virtual host names or IP addresses are being used.

ON
Indicates that the Web server is configured to use multiple virtual host names or IP addresses.

OFF
Indicates that the Web server is configured to use a single host name or IP address.

Note: The server will ignore this setting if the EnableFRCA directive is set off.

Example

```
FRCAVirtualHost   off
```

Initial configuration file setting

```
None
```

Program default setting

```
FRCAVirtualHost  auto
```

FRCAWLMParms — Specify parameters for Workload Management

Use this directive to specify the unique subsystem name, application environment name (AEName), and transaction class that will be used to classify the work performed by the Cache Accelerator under Workload Management (WLM).

The format of this directive is:

```
FRCAWLMParms subsystem_name [AEName][WLM_transaction_class ]
```

All parameters must be 8 characters or less.

The unique subsystem name is required. The transaction class and the AEName are optional. The Cache Accelerator activity is classified under WLM if you specify one of the following:

- The AEName, but not the transaction class
- The transaction class, but not the AEName
- The AEName, and the transaction class

If you specify neither the AEName nor the transaction class, no classification occurs. You can specify a wild card for the AEName, the transaction class, or both. The wildcard signifies an unspecified AEName, or transaction class. For more information on WLM parameters, see "ApplEnv - Specify application environment for workload management" on page 555. For tuning hints and tips, see "Monitoring and managing the Cache Accelerator" on page 187.

Note: The server will ignore this setting if the EnableFRCA directive is set off.

Example

```
FRCAWLMParms  FRCAHTTP WEBFRCA WEBFRCA
```

In this example, the unique subsystem name is `FRCAHTTP`, the AEName is `WEBFRCA`, and the transaction class is `WEBFRCA`.

Initial configuration file setting

```
FRCAWLMParms FRCAHTTP WEBFRCA WEBFRCA
```

Program default setting

```
None
```

ListenBacklog - Specify the number of listen backlog client connections for the server to carry

Use this directive to specify the number of listen backlog client connections you want the server to carry before sending connection refused messages to clients. This number depends upon the number of requests that your server can process in a few seconds and should not be set higher than the number the server can process before the clients timeout and abort the connection from their end. Requests involving secure transactions take longer, for instance, than client requests that do not require users to logon and give a password.

Note: If the ListenBacklog value is greater than the SOMAXCONN value supported by TCP/IP, SOMAXCONN will be used instead.

Example

```
ListenBacklog 100
```

Initial configuration file setting

```
ListenBacklog 128
```

LiveLocalCache - Specify whether the cache is updated when a cached file is modified

Use this directive to specify whether or not the cache is updated when a cached file is modified. Specify ON if you want users requesting a cached file to get the file with the latest updates. OFF is the high performance setting.

Initial configuration file setting

```
LiveLocalCache off
```

Program default setting

`LiveLocalCache off`

MaxActiveThreads - Specify the maximum number of active threads

Use the MaxActiveThreads directive to set the maximum number of threads that you want to have active at one time. If the maximum is reached, the server holds new requests until another request finishes and threads become available.

Prior to APAR PQ50195, the number of threads actually available for processing work equaled six threads less than the value specified on the MaxActiveThreads directive. The server used these six threads for DNS lookup. However, with the application of PQ50195, the Web server no longer needs the six threads for DNS lookup. The Web server returned them to the pool of available threads.

If you run your Web server in stand-alone mode, the Web server divides the threads between normal mode and Secure Sockets Layer (SSL) mode.

- If you specify normal mode on and SSL mode off, the server allocates all threads to normal mode.
- If you specify SSL mode on and normal mode off, the server allocates all threads to SSL mode.
- If you specify both SSL mode on and normal mode on, the server dynamically allocates the threads between both modes. Initially, the Web server evenly allocates the threads between the SSL mode work queue and the normal mode work queue. Once the work completes on a thread, the server decides if it should move the thread to the other work queue. However, work must complete for both SSL mode and normal mode for balancing to occur. If the incoming work requests are only for one of the two modes, the server only accesses threads on the active work queue. The threads on the inactive work queue are unused.

  For instance, if you have MaxActiveThreads set to 40, the Web server initially allocates 20 threads to the SSL mode work queue and 20 threads to the normal mode work queue. If only normal mode requests come into the server, then the Web server uses threads from the pool of 20 threads allocated to normal mode. The 20 threads allocated to SSL mode are unused.

If you run your Web server in scalable mode, the Web server divides the threads between four types of work queues: normal mode, SSL mode, and two scalable server work queues.

- If you specify normal mode on and SSL mode off, the server initially allocates the threads evenly between normal mode and the two scalable server work queues. Once work completes on a thread, the server decides if it should move the thread to another work queue
- If you specify SSL mode on and normal mode off, the server initially allocates the threads evenly between SSL mode and two scalable server work queues. Once work completes on a thread, the server decides if it should move the thread to another work queue
- If you specify both SSL mode on and normal mode on, the server initially allocates the threads evenly between SSL mode, normal mode, and the two scalable server work queues. Oncework completes on a thread, the server decides if it should move the thread to another work queue. However, work must complete for both SSL mode and normal mode for balancing to occur between all four queues. If the incoming work requests are only for SSL mode or

Normal mode, but not both, the server only accesses threads on the active work queues. The threads on the inactive work queue are unused.

For instance, if you have MaxActiveThreads set to 40, the Web server initially allocates ten threads to the SSL mode work queue, ten threads to the normal mode work queue, and 10 threads to each of the scalable server work queues. If only normal mode requests come into the server, then the Web server uses threads from the pool of 30 threads that are allocated to normal mode and the scalable server work queues. The 10 threads allocated to SSL mode are unused.

Give careful consideration to the number of threads you need and to which modes, SSL and normal, are active.

Note:
If the Web server is running in Scalable Server mode, a value of 100 or less is recommended; for Standalone mode, a value of 150 or less is recommended. If the percentage of static pages served is high, for example 60% or more, a higher setting may be needed. Experience has shown that a value greater than 200 can cause storage shortages. The setting of MaxActiveThreads must be lower than the z/OS UNIX BPXPRMxx setting for MAXTHREADTASKS.

Example

```
MaxActiveThreads   80
```

Program default setting

```
MaxActiveThreads   40
```

Initial configuration file setting

```
MaxActiveThreads   40
```

MaxContentLengthBuffer - Set the size of the buffer when computing content length

The Web server normally includes a content-length header field with every document it returns. If the content length is unknown, the server may buffer the content while computing the content length. The length is unknown if the content is from an MVS dataset, or when a request is proxied and the source server does not provide the content length header.

To prevent the overutilization of buffer resources by one request, use this directive to set the size of the buffer. If the value is exceeded, the buffering of the document will stop and the document will be returned without a content-length header field. The Web server gets this storage in 8K increments and will continue to buffer content until an 8K boundary is reached.

The value can be specified in kilobytes (K) or megabytes (M). Only one MaxContentLengthBuffer directive is allowed.

Example

```
MaxContentLengthBuffer 2 M
```

Initial configuration file setting

```
MaxContentLengthBuffer 100 K
```

Program default setting

```
MaxContentLengthBuffer 100 K
```

MaxPersistRequest - Specify the maximum number of requests to receive on a persistent connection

Use this directive to specify the maximum number of requests the server will allow on a persistent connection. When determining this number, be sure to consider the number of images used in your pages. Each image requires a separate request.

The format of this directive is:

```
MaxPersistRequest number
```

number is the number of requests the server will allow for a persistent connection.

Initial configuration file setting

```
MaxPersistRequest 5
```

Program default setting

```
MaxPersistRequest 10
```

PersistTimeout - Specify time to wait for the client to send another request

Use this directive to specify the amount of time the server should wait between client requests before cancelling a persistent connection.

The server uses a different timeout, the input timeout, to determine how long to wait for the client to send the first request after the connection is established. For more information on the input timeout, see "InputTimeout - Specify time allowed for the client to send a request" on page 564.

After the server sends its first response, it uses the persistent timeout to determine how long it should wait for each subsequent request before cancelling the persistent connection.

The format of this directive is:

```
PersistTimeout  time
```

time can be any valid time increment, but usually will be seconds or minutes.

Note: Persistent connections can be disabled by setting PersistTimeout to 0. When 0 is specified, each request is made over a new connection.

Examples

```
PersistTimeout 1 hour
PersistTimeout 2 mins 30 secs
PersistTimeout 15 seconds
```

Initial configuration file setting:

```
PersistTimeout 5 seconds
```

Program default setting:

```
PersistTimeout 5 seconds
```

QOS - Specify Quality of Service classification information you want to send to TCP/IP on each request

Use the QOS directive to specify the information that you want to send to TCP/IP on each request for quality of service classification. For information about Quality of Service, see the *z/OS Communications Server IP Configuration Guide*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

The format of this directive follows:

```
QOS  host | uri | hosturi
```

where:

host Classifies the request based on the host name header received in the request.

uri Classifies the request using the Uniform Resource Identifier (URI) received in the request.

hosturi
Classifies the request using both the host name header and the URI.

Example

```
QOS hosturi
```

Initial configuration file setting:

```
None
```

Program default setting:

```
None
```

ServerPriority - Specify the priority you want your server to have on your system

Use this directive to override the z/OS UNIX System Services default priority for scheduling the server process.

Valid values are:

-20 maximum z/OS UNIX System Services priority

0 default z/OS UNIX System Services priority

+19 minimum z/OS UNIX System Services priority

Example

```
ServerPriority   -20
```

Initial configuration file setting

```
ServerPriority   -10
```

UseACLs - Specify whether ACL files will be checked

Use this directive to specify whether the ACL files will be checked for file protection. Set this directive to NEVER or PROTECTONLY for better server performance. The format of the directive is:

```
UseACLs setting
```

The *setting* can have a value of always, protect only, or never.

always
The server will always look for an ACL file on every file request.

protectonly
The server will only look for an ACL file when the file request is for a file that is covered by a protection statement.

never
The server will never look for an ACL file on a file request.

Example

```
UseACLs protectonly
```

Initial configuration file setting

```
UseACLs protectonly
```

Program default setting

```
UseACLs protectonly
```

UseMetaFiles - Specify whether meta files will be used

Use this directive to specify whether the meta files used by the server. Set this directive to off for better server performance. The format of the directive is:

```
UseMetaFiles setting
```

The *setting* can have a value of on or off.

on The server will always use meta files.

off
The server will not use meta files.

Example

```
UseMetaFiles off
```

Initial configuration file setting

```
UseMetaFiles off
```

Program default setting

```
UseMetaFiles off
```

WLMSN- Enable the creation of request level Workload Management enclaves

Use the WLMSN directive to enable the creation of request level Workload Management (WLM) enclaves in standalone mode. This directive tells the server the subsystem name with which the Web server connects to WLM. Use this directive in conjunction with either the CLASSIFY environment variable, the Enclave directive, or both. For information on the CLASSIFY environment variable, see "Variables with values that can be set or created" on page 589. For information on Enclave, see "Enclave — Create and join a Workload Management enclave for the current request" on page 568.

Notes:

1. This feature is available in standalone mode only.
2. You cannot use FRCAWLMParms in conjunction with this feature.

The format of the directive follows:

```
WLMSN subsystem name
```

Example

```
WLMSN MYSUBSYS
```

Initial configuration file setting

None

Program default setting

None

Tuning directives

Appendix C. Data set naming reference

This appendix explains how to name z/OS data sets when you use the GWAPI MVSDS DLL Service. For more information on this service, see Appendix E, "GWAPI MVSDS DLL Service" on page 597.

Using certain qualifiers in data set names can affect how the Web server determines the data set's content type. The following table shows two examples:

QUALIFIER	CONTENT TYPE
HTML	HTML document
GIF	GIF image file

An HTML document should be stored in a data set whose fully-qualified name contains the qualifier **HTML**, for example:

```
WEBSRV.HTML.MYPAGE
```

Additional qualifiers can be used for language support or other purposes. The following example uses **ENU** as the qualifier:

```
WEBSRV.PAGES.HTML.ENU(MYPAGE)
```

In this example:

- The **PAGES** qualifier is included, but probably will not be recognized by the Web server as indicating the content type. Unrecognized qualifiers are ignored by the Web server.
- The **ENU** qualifier resolves to a U.S. English language document as defined by an AddLanguage directive.
- The **HTML** qualifier indicates an HTML document, as defined by an AddType or AddEncoding directive.

If you define a data set on a DD statement, then attributes will be determined from the data set qualifiers. For example, the DD statement is:

```
EXAMP DD DSN=MY.TXT
```

The URL is:

```
http://www.mvsexamp.com/MVSDS/DD:EXAMP(DOCA)
```

The content type attribute will be text/plain since MY.TXT ends with the qualifier TXT, and TXT is defined on the ADDTYPE directive as text/plain.

For more information on how qualifiers (suffixes) are processed, see "Multi-format processing - Define file extensions for multi-format processing" on page 520.Note that the Web server only does multi-format processing on z/OS UNIX files. It does not do multi-format processing on z/OS data sets.

Appendix D. Environment variables

Overview 583
Variables with values that are read-only 584
Variables with values that can be set or created . . 589
Server-Side Include variables 595

Overview

This appendix includes variables that you can use in GWAPI programs, CGI programs, and Server-Side Includes as part of processing a request. Many of these variables are defined on a request basis. Some are derived from settings in your httpd.envvars file, the z/OS UNIX System Services environment, Web server software levels, request headers, and the results of request processing.

- All variables can be used in GWAPI programs **except** for the following:

SSI_DIR
SSI_FILE
SSI_INCLUDE
SSI_PARENT
SSI_ROOT

- The following variables can be used in CGI programs:

AUTH_TYPE
CONTENT_ENCODING
CONTENT_LENGTH
CONTENT_TYPE
GATEWAY_INTERFACE
HTTP_ACCEPT
HTTP_COOKIE
HTTP_USER_AGENT
PATH_INFO
SERVER_SOFTWARE
PATH_TRANSLATED
QUERY_STRING
REFERER_URL
REMOTE_ADDR
REMOTE_USER
REQUEST_METHOD
SERVER_NAME
SERVER_PORT
SERVER_PROTOCOL

You can use the following SSL-related environment variables in CGI programs:

HTTPS
HTTPS_CLIENT_CERT_COUNTRY
HTTPS_CLIENT_CERT_ISSUER_COMMON_NAME
HTTPS_CLIENT_CERT_ISSUER_COUNTRY
HTTPS_CLIENT_CERT_ISSUER_LOCALITY
HTTPS_CLIENT_CERT_ISSUER_STATE_OR_PROVINCE
HTTPS_CLIENT_CERT_LEN
HTTPS_CLIENT_CERT_LOCALITY
HTTPS_CLIENT_CERT_ORGANIZATIO
HTTPS_CLIENT_CERT_ORG_UNIT
HTTPS_CLIENT_CERT_STATE_OR_PROVINCE
HTTPS_SESSION_ID_NEW

- You can use the following variables in server-side includes:

AUTH_TYPE
CONTENT_LENGTH
CONTENT_TYPE
DATE_GMT
DATE_LOCAL
DOCUMENT_NAME
REMOTE_USER
REQUEST_METHOD
SCRIPT_NAME
SERVER_ADDR
SERVER_NAME
SERVER_PORT

DOCUMENT_URI
GATEWAY_INTERFACE
HTTP_ACCEPT
HTTP_REFERER
HTTP_USER_AGENT
LAST_MODIFIED
PATH_INFO
PATH_TRANSLATED
QUERY_STRING
REMOTE_ADDR
REMOTE_HOST
SERVER_PROTOCOL
SERVER_ROOT
SERVER_SOFTWARE
SSI_DIR
SSI_FILE
SSI_INCLUDE
SSI_PARENT
SSI_ROOT

Variables with values that are read-only

The following variables contain values that can be extracted from a client request (read-only variables).

GWAPI Note: A return of HTTPD_READ_ONLY will result when attempting to change these variables using HTTPD_set.

ALL_VARIABLES
The server returns a list of environment variables, using newline as a separator. You can use ALL_VARIABLES when you convert a CGI to a GWAPI so that you can find out the environment variables that you used in your CGI. You can only use ALL_VARIABLES in the GWAPI version of your code, not the CGI version.. Following is a subset of the list of returned environment variables:

```
ACCEPT_RANGES ON
AUTH_STRING
CLIENT_ADDR 9.67.84.3
```

AUTH_STRING
The credentials sent with the request header AUTHORIZATION information

AUTH_TYPE
If the server supports client authentication and the script is protected, this environment variable contains the method used to authenticate the client, for example:

```
Basic
```

_BPX_USERID
The server returns the access control user ID associated with this request.

CACHE_HIT
Returns ″1″ if a document is found in cache, returns ″0″ otherwise.

CACHE_UPDATE
Returns ″1″ if cache information was updated successfully, returns ″0″ otherwise.

CLIENT_ADDR
IP address of the client, for example:

```
9.67.193.2
```

CLIENT_AUTH
State of SSL Client Authentication, for example, `On` or `Off`

CLIENT_HOST
Same as REMOTE_ADDR

CLIENTMETHOD
HTTP method used in the request

CLIENT_NAME
Host name of the machine making the request, for example:

```
joeblow
```

CONNECTIONS
Number of active connections the server has open, for example:

```
15
```

DATE_GMT
The current date and time in Greenwich Mean Time. The formatting of this variable is defined using the `config timefmt` directive.

DATE_LOCAL
The current date and local time. The formatting of this variable is defined using the `config timefmt` directive.

DOCUMENT_NAME
This is the name of the topmost document. If the HTML was generated by a CGI, this will contain the name of the CGI.

DOCUMENT_ROOT
The path to the document or script as defined by Pass rules

DOCUMENT_URI
That part of the URL that specifies the path and file name to a resource on the server. Example:

```
/software/webservers/httpservers/doc/v51/tdoc1.htm
```

DOCUMENT_URL
The Uniform Resource Locator, for example:

```
http://www.urlexamp.com/Admin/lgmast.gif
```

FSCP Returns the value specified on the DefaultFsCp configuration directive

GATEWAY_INTERFACE
Contains the version of CGI or GWAPI that the server is using, for example: `CGI/1.1` or `GWAPI/1.2`

HTSERVLETCODE
Server-side includes servlet class name.

HTSERVLETCODEBASE
The URI location of the server-side includes servlet code.

HTSERVLETINITPARAMS
Initial value pairs for the server-side includes servlet.

HTSERVLETNAME
Server-side includes servlet name.

HTSERVLETREGPARAMS
Server-side includes parameter value pairs.

HTTP_AUTHORIZATION
Authorization header information

HTTP_PROXY_AUTHORIZATION
Proxy Authorization header information

HTTP_REFERER
Request header information

HTTPS
This is valid only if security is active and a valid client certificate is available.

HTTPS_KEYSIZE
The value of security key size

[128]

HTTPS_CLIENT_CERT
The client certificate body

HTTPS_CLIENT_CERT_COMMON_NAME
From the certificate body

HTTPS_CLIENT_CERT_COUNTRY
From the certificate body

HTTPS_CLIENT_CERT_LOCALITY
From the certificate body

HTTPS_CLIENT_CERT_ORGANIZATION
From the certificate body

HTTPS_CLIENT_CERT_ORG_UNIT
From the certificate body

HTTPS_CLIENT_STATE_OR_PROVINCE
From the certificate body

HTTPS_CLIENT_CERT_LEN
Length of client certificate body

HTTPS_CLIENT_CERT_SERIAL_NUM
The serial number from the certificate body

HTTPS_CLIENT_CERT_ISSUER_COMMON_NAME
From the certificate body

HTTPS_CLIENT_CERT_ISSUER_COUNTRY
From the certificate body

HTTPS_CLIENT_CERT_ISSUER_LOCALITY
From the certificate body

HTTPS_CLIENT_CERT_ISSUER_ORGANIZATION
From the certificate body

HTTPS_CLIENT_CERT_ISSUER_ORG_UNIT
From the certificate body

HTTPS_CLIENT_CERT_ISSUER_STATE_OR_PROVINCE
From the certificate body.

HTTPS_SESSION_ID
The current session ID

HTTPS_SESSION_ID_NEW
This environment variable indicates that the Secure Sockets Layer (SSL) connection renegotiated new session keys and is not using any session information from a previous request. Possible values are 1 or 0.

INIT_STRING
The string specified on the ServerInit directive

LDAP:variable
This reads the LDAP variable specified.

```
LDAP:CURRENT:HOST
```

NETCP
Returns the value specified on the DefaultNetCp configuration directive.

PICS_PATHNAME
Path and file name of the document to be rated

PICS_SERVICENAME
Name of the PICS rating service, for example:

```
Ratings USA
```

PICS_SITENAME
Example:

```
http://www.ratings.com/
```

PROXY_ACCESS
Defines whether the request is a proxy request, for example:

```
NO
```

REQUEST_CONTENT_LENGTH
This is the same as CONTENT_LENGTH.

REQUEST_CONTENT_TYPE
See CONTENT_TYPE to find out under what conditions REQUEST_CONTENT_TYPE, CONTENT_TYPE, and HTTP_CONTENT_TYPE are the same and different.

RESPONSE_CONTENT_LENGTH
The value for RESPONSE_CONTENT_LENGTH comes from the Content-Length header set by the server if the header is available. RESPONSE_CONTENT_LENGTH may or may not have the same value as REQUEST_CONTENT_LENGTH.

RESPONSE_CONTENT_TYPE
The value for RESPONSE_CONTENT_TYPE comes from the Content-Type header set by the server if the header is available. RESPONSE_CONTENT_TYPE may or may not have the same value as REQUEST_CONTENT_TYPE.

REQUEST_METHOD
Contains the method (as specified with the METHOD attribute in an HTML form) used to send the request, for example:

```
GET or POST
```

REQHDR
The server returns a list of the request headers separated by newline.

REFERER_URL
Contains the last URL location of the browser if the browser sends a referer header, for example:

```
http://www.acme.com/homepage
```

SDH Socket descriptor for the request

SERVER_ADDR
Local IP address of the server

SERVER_CFG_PORT
Contains the port number the server is accepting http requests on, for example, 80; from the PORT configuration directive or the command line option -p.

SERVER_CFG_SSLPORT
Contains the port number the server is accepting SSL requests on, for example, 443; from the PORT configuration directive or the command line option -sslport.

SERVER_LEVEL
The release level of the currently running server

SERVER_NAME
Contains the server host name or IP address of the server, for example:

```
www.ibm.com
```

SERVER_PORT
Contains the port number of the server to which the client request was sent, for example:

```
80
```

SERVER_PROTOCOL
Contains the name and version of the protocol used to make the request, for example:

```
HTTP/1.1
```

SERVER_ROOT
Directory where the server program is installed

SERVER_SOFTWARE
Contains the name and version of the server, for example:

```
HTTP Server/Version 5.3
```

SERVER_STATE
Current process step for the request (for example, "PREEXIT")

SSI_DIR
The path of the current file relative to SSI_ROOT. If the current file is in SSI_ROOT, this value is "/".

SSI_FILE
The file name of the current file.

SSI_INCLUDE
The value used in the include command that retrieved this file. This value is not defined for the topmost file.

SSI_PARENT
The path and file name of the includer, relative to SSI_ROOT.

SSI_ROOT
The path of the topmost file. All include requests must be in this directory or a child of this directory.

SSLMODE
Whether Secure Sockets Layer (SSL) is enabled. This variable is read only. If you attempt to modify it using HTTPD_set, the server will return an HTTPD_READ_ONLY error. For example:

```
ON
```

TOTALKBYTES
Total number of kilobytes the HTTP Server has served, for example:

```
1000
```

TOTALTRANSACTIONS
Total number of transactions the HTTP Server has served. For example:

```
300
```

WQ_APPLENV
Workload Management application environment assigned by the -AE option

WQ_STATE
Workload Management work queue state, for example, HTTPD, WQ_Daemon, and WQ_Server.

WQ_SUBSYS
Workload Management subsystem name assigned by the -SN option.

Variables with values that can be set or created

The following variables contain values you can set or create when processing a client request.

ACCEPT_RANGES
Used to accept ranges other than bytes

CGI_variable
Returns the value of the user-defined CGI variable, where variable isuser-defined from an HTTPD_set() command (for example, CGI_MYVARIABLE). This CGI_ variable is appended to the list of environment variables sent to a CGI program.

CLASSIFY
Creates and joins an enclave for the current request, based on the classification information sent to the Web server from Workload Management (WLM). Use this environment variable in conjunction with the WLMSN directive. For information on WLMSN, see "WLMSN- Enable the creation of request level Workload Management enclaves" on page 578.

If a request matches a URI template on an Enclave directive, the Web server opens the enclave under the Enclave directive classification. The Web server opens an enclave that uses the CLASSIFY environment variable only if the request does not match a URI template on an Enclave directive. The enclave remains open under the specified classification until request processing completes. Only one enclave at a time is open. For more information on the Enclave directive, see "Enclave — Create and join a Workload Management enclave for the current request" on page 568.The structure passed in the classification call has the following format:

```
typedef struct _classify_info {
    char *classify_tc;
    char *classify_tn;
    char *classify_userid;
    char *classify_subsystem_parm;
} HTClassify_info;
```

where:

- **classify_tc** is the transaction class (length 8).
- **classify_tn** is the transaction name (length 8).
- **classify_userid** is the user ID (length 7).
- **classify_subsystem_parm** is the subsystem parameters (length 255).

You are responsible for allocating the storage for the structure and passing the pointer to the structure in the value field. Set any fields to NULL that you do not want to classify. Set the value_length value to the total size of the above structure.

Note: This variable is not available for extraction (HTTPD_extract).

Related information: For more information on these classification fields, see the __server_classify() function in the *z/OS C/C++ Run-Time Library Reference*. To access this book on the Web, go to the z/OS Book Server Web site at URL: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`.

CLASSIFY_USERID
The user ID that is put in the WLM classify structure during the WLM put work request. This user ID can only be set and extracted during the WLMClassify pre-exit step. For additional information, see "WLMClassify - Customize the WLM pre-exit step" on page 478.

CLIENT_PROTOCOL
Name and version of the protocol the client is using to make the request, for example:

```
HTTP
```

CONTENT_CHARSET
Character set of the response for text/*, for example:

```
US ENGLISH
```

CONTENT_TYPE_PARAMETERS
Other MIME attributes but not the character set

CONVERT_REQUEST_BODY
Only for z/OS EBCDIC issues; specifies if the ASCII to EBCDIC conversion is done on the request body. Available settings are YES, NO, and AUTO. The default is NO. If AUTO is set, the server tries to make the decision in the same way CGI requests are converted. For example:

```
AUTO
```

CONTENT_ENCODING
Specifies the encoding used in the document. For example:

```
x-gzip
```

CONTENT_LENGTH
CONTENT_LENGTH, HTTP_CONTENT_LENGTH, and REQUEST_CONTENT_LENGTH all contain the same value. The server uses the Content-Length header sent by the client to set the value. If the client does not send the Content-Length header, all three variables are null.

When the client sends the method of POST, then CONTENT_LENGTH, HTTP_CONTENT_LENGTH, and REQUEST_CONTENT_LENGTH each contain the number of characters of data. Servers typically do not send an end-of-file flag when they forward the information by using standard in. You can use the value of CONTENT_LENGTH, HTTP_CONTENT_LENGTH, or REQUEST_CONTENT_LENGTH to determine the end of the input string. For example:

```
7034
```

CONTENT_TYPE
CONTENT_TYPE, HTTP_CONTENT_TYPE, and REQUEST_CONTENT_TYPE all contain the same value if the client sends the Content-Type header. If the client does not send a Content-Type header,

the value for CONTENT_TYPE and HTTP_CONTENT_TYPE comes from the Content-Type header set by the server if the header is available. REQUEST_CONTENT_TYPE will be null.

When the client sends the method of POST, CONTENT_TYPE, HTTP_CONTENT_TYPE, and REQUEST_CONTENT_TYPE contain the type of data included. You can create your own content type in the server configuration file and map it to a viewer. For example:

```
application/x-www-form-urlencoded
```

ERRORINFO
Specifies the error code to determine the error page, for example:

```
401
```

EXPIRES
Defines the expiration for documents stored in a proxy's cache

HTTP_ACCEPT
Contains the list of MIME types the browser accepts. For example:

```
text/html
```

HTTP_ACCEPT_RANGES
This is the same as ACCEPT_RANGES.

HTTP_CONTENT_CHARSET
This is the same as CONTENT_CHARSET.

HTTP_CONTENT_ENCODING
This is the same as CONTENT_ENCODING.

HTTP_CONTENT_LENGTH
This is the same as CONTENT_LENGTH.

HTTP_CONTENT_TYPE
This is the same as CONTENT_TYPE.

HTTP_COOKIE
Contains the cookie sent with this request; used to communicate state information. For example:

```
CustomerNumber=HJ68944
```

You can find details about cookies at http://www.netscape.com/newsref/std/cookie_spec.html.

HTTP_EXPIRES
This is the same as EXPIRES.

HTTP_LAST_MODIFIED
This is the same as LAST_MODIFIED.

HTTP_REASON
Sets the reason string in the HTTP response header

HTTP_RESPONSE
Sets the response code in the HTTP response header

HTTP_USER_AGENT
Contains the name of your Web browser. For example:

```
Mozilla/4.7 [en] (WinNT; U)
```

LAST_MODIFIED
The date and time that the current file was last modified. If the current file is a script, then this variable is set to the current date and time. If you use

this variable in a server-side include, the formatting of this variable is defined using the `config timefmt` directive. Example:

```
Wed, 31 May 2000 15:00:17 GMT
```

LOCAL_VARIABLES
All the user-defined variables

PASSWORD
For Basic authentication, contains the decoded password, for example:

```
password
```

PATH If a request is for a static file, PATH is the physical path to the file that you want the server to return (*/path/file*). If a request is for an application function that you specified on a Service directive, PATH consists of the physical path to your compiled program, along with the function name (*/path/file:function*). If a request is for a CGI, PATH is the physical path to your CGI program (*/path/file*).

Note: PATH is **not** the same as the z/OS UNIX System Services PATH variable.

Examples

httpd.conf includes the following directives:

```
Exec    /cgi-bin/*  /u/user125/cgia/*
Pass    /SAMPFILE/* /GWAPIT2/*
Map     /SAMP/*     /GWAPIT/*
Service /GWAPIT/*   /u/user125/regapi/regapi:myservice/*
```

Example 1: The URL is:

```
http://www.test.com/SAMPFILE/pwdchanged.html
```

The URI matches on a Pass directive, and PATH is

```
/GWAPIT2/pwdchanged.html
```

Example 2: The URL is:

```
http://www.test.com/GWAPIT/pwdchanged.html
```

The URI matches on a Service directive, and PATH is

```
/u/user125/regapi/regapi:myservice
```

Example 3: The URL is:

```
http://www.test.com/cgi-bin/pass/my_cgi_program
```

The URI matches on an Exec directive, and PATH is

```
/u/user125/cgia/pass/my_cgi_program
```

PATH_INFO
If a URI matches on a Service directive, PATH_INFO contains all the path information of the URI that would be represented by the wild card (*) on the */path/file:function** template of the Service directive. PATH_INFO is null if the URI matches on a Service directive, but there is no wild card on */path/file:function*. If a URI matches on a Pass directive for a static file, PATH_INFO is null. If a URI matches on an Exec directive for a CGI, PATH_INFO is that part of the URI that consists of input to the CGI program. (PATH_INFO is *input* in */path/file/input*).

Examples

httpd.conf includes the following directives:

```
Exec    /cgi-bin/*  /u/user125/cgia/*
Pass    /SAMPFILE/* /GWAPIT2/*
Map     /SAMP/*     /GWAPIT/*
Service /GWAPIT/*   /u/user125/regapi/regapi:myservice/*
Service /GWAPIT2*   /u/user125/regapi2/regapi:myservice*
```

Example 1: The URL is:

`http://www.test.com/GWAPIT/testa/test_path_info`

The URI matches on a Service directive, PATH is

`/u/user125/regapi/regapi:myservice`

and PATH_INFO is

`/testa/test_path_info`

Example 2: The URL is:

`http://www.test.com/GWAPIT2/testb/test_path_info`

The URI matches on a Service directive, PATH is

`/u/user125/regapi2/regapi:myservice`

and PATH_INFO is

`/testb/test_path_info`

Example 3: The URL is:

`http://www.test.com/cgi-bin/testc_program/testc_input`

The URI matches on an Exec directive, PATH is

`/u/user125/cgia/testc_program`

and PATH_INFO is

`/testc_input`

PATH_TRANSLATED

In order to obtain the value for PATH_TRANSLATED, the Web server takes the value of PATH_INFO and goes through the Pass and Map directives looking for matches. The value of PATH_INFO will ultimately match on a Pass directive if you at least have a request template of /* on a Pass directive. If there is a match on the request template of a Map directive, the Web server takes the new request string of the Map directive and compares it to subsequent directives until it matches on a Pass directive. When there is finally a match on a request template for a Pass directive, PATH_TRANSLATED contains the physical path to the file (*/path/file*).

Examples

httpd.conf includes the following directives:

```
Exec    /cgi-bin/*  /u/user125/cgia/*
Pass    /SAMPFILE/* /GWAPIT2/*
Map     /SAMP/*     /GWAPIT/*
Map     /SAMP2/*    /GWAPIT3/*
Pass    /GWAPIT3/*  /u/user125/target/*
Service /GWAPIT/*   /u/user125/regapi/regapi:myservice/*
```

Example 1: The URL is:

`http://www.test.com/GWAPIT/SAMPFILE/test.html`

PATH_INFO is
`/SAMPFILE/test.html`

PATH_TRANSLATED is
`/GWAPIT2/test.html`

Example 2: The URL is:
`http://www.test.com/GWAPIT/SAMP2/test.html`

PATH_INFO is
`/SAMP2/test.html`

PATH_TRANSLATED is
`/u/user125/target/test.html`

PICS_LABEL
PICS label information

PPATH
If the URI matches on the request template of a Map directive, PPATH is the same as the new request string. If the URI of a request does not match on the request template of a Map directive, PPATH is the same as the URI.

Examples

httpd.conf includes the following directives:

```
Exec    /cgi-bin/*  /u/user125/cgia/*
Pass    /SAMPFILE/* /GWAPIT2/*
Map     /SAMP/*     /GWAPIT/*
Service /GWAPIT/*   /u/user125/regapi/regapi:myservice/*
```

Example 1: The URL is:
`http://www.test.com/SAMP/pwdchanged.html`

PATH is
`/u/user125/regapi/regapi:myservice`

PPATH is
`/GWAPIT/pwdchanged.html`

Example 2: The URL is:
`http://www.test.com/SAMPFILE/pwdchanged.html`

PATH is
`/GWAPIT2/pwdchanged.html`

PPATH is
`/SAMPFILE/pwdchanged.html`

PROXY_CONTENT_LENGTH
Content-Length header of the proxy request made through HTTPD_proxy. When information is sent with the method of POST, this variable contains the number of characters of data. Servers typically do not send an

end-of-file flag when they forward the information using stdin. If needed, you can use the CONTENT_LENGTH value to determine the end of the input string. For example:

```
7034
```

PROXY_CONTENT_TYPE
Content-Type header of the proxy request made through HTTPD_proxy. When information is sent with the method of POST, this variable contains the type of data included. You can create your own content type in the server configuration file and map it to a viewer. For example:

```
application/x-www-form-urlencoded
```

PROXY_METHOD
Method for the request made through HTTPD_proxy

QUERY_STRING
When information is sent using a method of GET, this variable contains the information in a query that follows the ?. This information must be decoded by the CGI program. For example:

```
NAME=Eugene+T%2E+Fox&ADDR=etfox%7Cibm.net&INTEREST=xyz
```

REMOTE_ADDR
Contains the IP address of the client, if available. For example:

```
9.23.06.8
```

REMOTE_HOST
Contains the host name of the client, if available. For example:

```
user1.raleigh.ibm.com
```

REMOTE_USER
If the request requires authentication, this variable contains the user ID given for authentication. This is not necessarily the same user ID under which the request is running. (_BPX_USERID contains the ID under which the request is running.)

SCRIPT_NAME
URI of the request

SSI_GLOBAL_VARS
All the SSI global variables.

SSI_variable
Reads the user-defined server-side includes variable. variable is the user-defined name.

URI Same as DOCUMENT_URI

URL Same as DOCUMENT_URL

USERNAME
Same as REMOTE_USER

Server-Side Include variables

Use National Computer Security Association (NCSA) tags to print the value of any of the server-side include variables in this appendix. The values will be printed on a Web page. For example, to print the name of the server, use the following NCSA tag:

```
<!--#echo var=SERVER_NAME -->
```

The following variables can only be used in server-side includes. All variables are read-only.

SSI_DIR
The path of the current file relative to SSI_ROOT. If the current file is in SSI_ROOT, this value is "/". This variable is read only. If you attempt to modify it using HTTPD_set, the server will return an HTTPD_READ_ONLY error.

SSI_FILE
The file name of the current file. This variable is read only. If you attempt to modify it using HTTPD_set, the server will return an HTTPD_READ_ONLY error.

SSI_INCLUDE
The value used in the include command that retrieved this file. This value is not defined for the topmost file. This variable is read only. If you attempt to modify it using HTTPD_set, the server will return an HTTPD_READ_ONLY error.

SSI_PARENT
The path and file name of the includer, relative to SSI_ROOT. This variable is read only. If you attempt to modify it using HTTPD_set, the server will return an HTTPD_READ_ONLY error.

SSI_ROOT
The path of the topmost file. All include requests must be in this directory or a child of this directory. This variable is read only. If you attempt to modify it using HTTPD_set, the server will return an HTTPD_READ_ONLY error.

Appendix E. GWAPI MVSDS DLL Service

Preparing the MVSDS DLL Configuration File . . 597
Specifying the MVSDS DLL configuration directives 597
Examples 598
Enable MVSDS DLL preloading of z/OS data sets using GWAPI directives in the server configuration file 598
Example 598
Accessing Web contents from z/OS data sets 599
Performance issues when accessing z/OS data sets 600
Return codes from MVSDS functions 601

This GWAPI service allows you to access Web contents stored in z/OS data sets. Data sets to be preloaded are specified in a separate configuration file which is specific to the MVSDS DLL. The default MVSDS DLL configuration file is `/etc/mvsds.conf`. Preloading of z/OS data sets is suggested for frequently accessed Web content. The Web server preloads data sets during initialization of the Web server and on restart of the Web server. When a request comes into the Web server for a preloaded data set, the Web server reads the preloaded data set. Note that MVS data sets can be accessed without preloading them. Each time the Web server processes a request for a data set that is not preloaded, the Web server reads the data set directly.

Preparing the MVSDS DLL Configuration File

The default MVSDS DLL configuration file is `/etc/mvsds.conf`.

You can specify an alternate location and file name by using the MVSDS_CFG LE environment variable. Note that this method for specifying an alternate configuration file has precedence over the program default. To set environment variables, see "IMWHTTPD program" on page 400 and the *LE Programming Guide*.

You can also specify the MVSDS DLL configuration file on the ServerInit directive in the server configuration file.

If no configuration file is found, no initial configuration of the MVSDS DLL occurs and no MVS data sets will be preloaded at server initialization.

Specifying the MVSDS DLL configuration directives

The MVSDS DLL configuration file (by default, `/etc/mvsds.conf`) contains LOAD directives, one per line, in the following format:

```
LOAD datasetname
```

datasetname
Specifies the z/OS data set name to be preloaded. A *datasetname* can be:

- A JCL DD name (DDN), optionally followed by a PDS member name. If a PDS member name is included, the DDN must resolve to a PDS.
- A fully-qualified name, optionally followed by a PDS member name, and enclosed in single quotes.
- A partially-qualified name, optionally followed by a PDS member name to which the MVS user ID (for example, WEBSRV) gets prepended as a high-level qualifier (no quotes). Note that the MVS user ID used in any fully-qualified name is the user ID the server is running under.

The format for the *datasetname* is:

```
[JCL DD|fully-qualified name|partially-qualified name] [(member)]
```

name
Specifies a sequence of one or more qualifiers, each containing eight or less characters, separated by periods.

member
Specifies a member name of no more than eight characters.

Note: If you are specifying a fully-qualified name, single quotes are required around the entire *name* and *member* string.

Examples

The following shows LOAD directive examples:

```
LOAD 'WEBSRV.HTML.ENU.PAGES'
LOAD IMAGES.GIF.TREES
LOAD 'WEBSRV.WAV.SOUNDS'
LOAD DD:SALES(JANUARY)
```

Note: Whenever a PDS is specified in a LOAD directive, the PDS directory and all of the members are preloaded.

Configuration of the MVSDS DLL preload data sets is also supported through the Administration and Configuration forms.

Enable MVSDS DLL preloading of z/OS data sets using GWAPI directives in the server configuration file

Preloading of z/OS data sets is suggested for frequently accessed Web content. The Web server preloads data sets during initialization of the Web server and on restart of the Web server. When a request comes into the Web server for a preloaded data set, the Web server reads the preloaded data set.

To instruct the server to preload z/OS data sets defined in the MVSDS DLL configuration file, you must implement three of the GWAPI directives in the server configuration file (by default, /etc/httpd.conf). ServerInit and ServerTerm are optional directives for preloading data sets. However, if a ServerInit is specified, a ServerTerm must be specified. It is suggested that you use this mechanism to preload data sets which contain frequently accessed Web contents. The three GWAPI directives used are ServerInit, Service, and ServerTerm. The format for the ServerInit directive is:

```
ServerInit /path/file:function_name [INIT_STRING]
```

Example

To use the default location of **/etc/mvsds.conf**, enter:

```
ServerInit /usr/lpp/internet/bin/mvsds.so:mvsdsInit
```

The INIT_STRING parameter on the ServerInit directive can be used to specify an alternate MVSDS DLL initialization file, for example:

```
ServerInit /bin/mvsds.so:mvsdsInit /u/WEBSRV/config/mymvsds.conf
```

The value of the INIT_STRING can be extracted like any other server CGI variable. If no INIT_STRING is defined, an empty string is returned.

The ServerTerm directive is used to clean up the DLL during the server restart to avoid memory leaks. See Appendix B, “Configuration directives” on page 417 for information about the ServerInit, Service, and ServerTerm directives.

The format of the ServerTerm directive is:

```
ServerTerm /path/file:function_name
```

An example of this directive is:

```
ServerTerm /bin/mvsds.so:mvsdsTerm
```

Accessing Web contents from z/OS data sets

The only directive you need if you are not preloading an MVS data set is the Service directive. You must have the Service directive specified in the server configuration file to activate the GWAPI MVSDS DLL Service. The format for this directive is:

```
Service request-template /path/file:function_name [IP_address_template]
```

Here are examples of the Service directive:

```
Service /MVSDS* /usr/lpp/internet/bin/mvsds.so:mvsdsGet*
Service /MYDATA* /usr/lpp/internet/bin/mvsds.so:mvsdsGet*
Service /SALES* /usr/lpp/internet/bin/mvsds.so:mvsdsGet*
```

Web contents stored in the z/OS data sets can be accessed by using a URL in the following format:

```
http://hostname/request_template/datasetname
```

hostname
Specifies the domain name or IP address returned to the client.

request_template
Specifies to the server that the file specified by the *datasetname* following the *request_template* is an MVS data set, not an HFS file or directory. The *request_template* can be anything meaningful to you that does not conflict with or override other matching directives. This should correspond to the *request_template* specified on the Service directive.

The following are examples of URL's containing data set names:

```
http://www.mvs105.tcp.ibm.com/MVSDS/'WEBSRV.PAGES.HTML.HOME'
http://www.mvs105.tcp.ibm.com/MVSDS/DD:PAGES(OTHERPAG)
http://www.mvs105.tcp.ibm.com/MVSDS/MYPAGES.IMAGES.GIF(NATURE)
```

Where MVSDS is the *request_template*. The first example specifies a fully-qualified name, the second example specifies a JCL DD reference to a PDS whose member OTHERPAG is to be retrieved, and the third specifies a partially-qualified name to which the MVS user ID will be prepended.

datasetname
Specifies the name of the MVS data set.

The *datasetname* specified in the URL can be one of the following:

'THIS.THAT.THEOTHER'
Specifies, in quotes, a fully-qualified data set name. The specified data set is opened directly.

'THIS.THAT.THEOTHER(MEMBER)'
Specifies, in quotes, a fully-qualified data set name represents a PDS member. The specified member is opened directly.

THIS.THAT.THEOTHER
Specifies a partially-qualified data set name to which a high level qualifier (hlq) must be prepended. In the z/OS environment, the MVS user ID is

prepended to the specified name to produce the fully-qualified name 'USERID.THIS.THAT.THEOTHER'. which is opened.

THIS.THAT.THEOTHER(MEMBER)
Specifies a partially-qualified PDS member. The hql (MVS user ID) is prepended to produce the fully-qualified name 'USERID.THIS.THAT.THEOTHER(MEMBER)', which is opened.

DD:THISNAME
Specifies that the actual data set or PDS member name is retrieved from the associated JCL DD statement and is then opened.

DD:THISNAME(MEMBER)
Specifies that the PDS name is retrieved from the associated JCL DD statement and the MEMBER specified is opened.

If a member name is not specified using the (MEMBER) form, the MVSDS DLL service determines whether the named data set is a PDS or a sequential data set. The contents of the data set get written back to the HTTP Server, depending on the type of data set:

- An HTML directory listing is returned for a PDS
- The data set attributes (content type, content encoding, language encoding, etc.) are evaluated and established in the server environment according to a predefined naming convention. The contents are then returned to the server ″as is'' for a sequential data set or a PDS member.

Note: MVSDS will not be able to serve any data set name that contains a pound sign (#) or a member name that contains a pound sign that is explicity requested in a URL. This is because of the nature of http, where a pound sign is treated as a terminating character. However, a member list for a partitioned data set that is generated by MVSDS may contain member names that contain pound signs.

Note: MVSDS requires that a fully qualified dataset be enclosed in quotes. Since some browsers read the file name from the URL, they may interpret the final qualifier with the quote as a file extension. For example, if the fully qualified dataset is 'THIS.THAT.ABC', the browser may interpret the file extension as .ABC' including the final quote. In some cases, such as executable applets, the file extension of .ABC' may not be recognized. The unrecognized file extension causes the browser to attempt to store the file instead of executing or displaying it on the browser.

For more information on MVSDS, see Appendix C, "Data set naming reference" on page 581. This appendix discusses how to name data sets. It also discusses how to relate data set qualifiers to content type, language support, and so on.

Performance issues when accessing z/OS data sets

When the Web server accesses z/OS data sets, dynamic allocation occurs as part of system services. Dynamic allocation occurs serially. Delays can occur in the dynamic allocation process, for example, when retrieving migrated data sets. Consequently, any request that goes through dynamic allocation while another request is delayed will be delayed too.

If you preload a data set or use a DD statement to reference a data set, you can avoid dynamic allocation when a client requests the data set. In these cases, dynamic allocation occurs when the Web server initializes.

Return codes from MVSDS functions

The valid return codes from MVSDS functions are:

mvsdsInit

Return code	Keyword	HTTP response code
HTTP_OK		200
HTTP_SERVER_ERROR	scriptio	500

mvsdsTerm

Return code	Keyword	HTTP response code
HTTP_OK		200

mvsdsGet

Return code	Keyword	HTTP response code
HTTP_OK		200
HTTP_BAD_REQUEST	badrequest	400
HTTP_FORBIDDEN	openfailed	403
HTTP_NOT_FOUND	notfound	404
HTTP_SERVER_ERROR	scriptio	500

Appendix F. Messages

z/OS LookAt online message facility 603
Support services and resources 604
Message catalog errors 604
Overview of IMW Messages 604
Message severity 604
Message ID ranges and types 604
IMW0001E-0572E: IMWHTTPD Messages 605
Explanation of errno and errno2 codes in messages 605
Message descriptions 605
IMW2000E-2026E: Proxy Server Messages 658
IMW3501I-3546E: CONSOLE Messages 661
Explanation of process descriptor in messages 661
Message descriptions 661
IMW3701E-3730E: HTCounter Program Messages 666
IMW4000E-4018E: HTIMAGE Messages 669
IMW5001E-5010E: HTADM Messages 670
IMW6102I-6805E: SSL Security Messages 672

z/OS LookAt online message facility

LookAt is an online facility that enables you to look up explanations for z/OS messages, system abends, and some codes. Use of the LookAt facility to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access the LookAt facility from the Internet at the following uniform resource locator (URL):

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html`

or from anywhere in the z/OS environment where you can access a Time Sharing Option (TSO) command line (for example, TSO prompt, Interactive System Productivity Facility (ISPF), z/OS UNIX System Services running OMVS).

To find a message explanation on the Internet, go to the LookAt Web site and simply enter the message identifier (for example, `IMW0541E` or `IMW*`). You can select a specific release to narrow your search.

You can also download code from the *z/OS Collection*, SK3T-4269, or the LookAt Web site so you can access LookAt from a PalmPilot (Palm VIIx suggested).

To use LookAt as a TSO command, you must have the LookAt facility installed on your host system. You can obtain the LookAt code for TSO from a disk on your *z/OS Collection*, SK3T-4269, or from the LookAt Web site. To obtain the code, go to the LookAt Web site at `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html`, and do the following:

1. Click **News**.
2. Scroll to **Download LookAt Code for TSO and VM.**
3. Click the **ftp** link, which takes you to a list of operating systems. Click the appropriate operating system. Then click the appropriate release.
4. Find the **lookat.me** file and follow its detailed instructions.

To find a message explanation from a TSO command line, simply enter: `lookat` *`message-id`*. The LookAt facility displays the message explanation for the message requested.

Note: Some messages have information in more than one book. For such messages, the LookAt facility prompts you to choose the book to open.

Support services and resources

For information and links to IBM support services and resources on the Web, refer to the Support section of the *WebSphere Troubleshooter for z/OS and OS/390* .

To link to the *Troubleshooter*, go to URL:

`http://www.ibm.com/software/webservers/httpservers/troubleshooter.html`

Message catalog errors

When the Web server loads the message catalog, it verifies that the message catalog is the correct version. If the version is incorrect, the Web server issues all messages with the following text instead of the text normally used in the messages. You must have APAR PQ50699 applied to receive this diagnostic message.

Message catalog version mismatch, version *version*

The *version* indicates the actual level of the message catalog that the Web server loaded. If blank, the version of the message catalog is older than the version of the message catalog associated with APAR PQ50699. If a mismatch exists between the version of the loaded message catalog and the version the server requires, the server terminates. The completion code is 211.

If the Web server is unable to load the message catalog, it writes the following message, and terminates with completion code 211.

Error opening message catalog.

Overview of IMW Messages

Message severity

The alphabetic character at the end of each message ID indicates the message's severity:

I Informational message

E Recoverable error

W Warning

S Serious error

Message ID ranges and types

Each server component has a range of message IDs:

Table 9. Message Ranges for the Server Components

Message ID Range	Component
IMW0001-IMW2000	IMWHTTPD messages
IMW2000-IMW2500	Proxy Server messages
IMW3501-IMW3700	CONSOLE Messages
IMW3701-IMW3999	HTCounter Program Messages
IMW4000-IMW5000	HTIMAGE messages
IMW5001-IMW6000	HTADM messages
IMW6100-IMW6900	SSL Security Messages

IMW0001E-0572E: IMWHTTPD Messages

Explanation of errno and errno2 codes in messages

Many IMWHTTPD messages include z/OS UNIX System Services return (*errno*) and reason (*errno2*) codes that can provide additional information about the cause of a failure or error.

For an explanation of these codes, see the following sections in the *z/OS UNIX System Services Messages and Codes* book:
- For errno codes, Return Codes Listed by Value
- For errno2 codes, Reason Codes Listed by Value

To access the *z/OS UNIX System Services Messages and Codes* book on the Web, go to the z/OS Book Server at URL:

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

Message descriptions

IMW0001E Cannot get hostname. Icon URLs will not have host name.

Explanation: The server cannot determine its own hostname using gethostname(). The HTML generated by the server in response to directory list requests contains relative URLs for the file type icons (.gif) preceding each file in the list. The server normally generates absolute URLs that contain its own network name.

User Response: Correct your system or name server configuration so gethostname() returns the name of the system that your server is running on.

IMW0002E Cannot open the configuration file: *name***.**

Explanation: Indicates configuration file *name* cannot be read.

User Response: Verify the name of the configuration file specified in the -r parameter of the server job stream. The default file is /etc/httpd.conf. If the named file exists, verify that your server has appropriate permissions to read the file.

IMW0003E Configuration not loaded due to errors.

Explanation: The configuration file contains errors and was not loaded. See the server error log or trace log for details.

User Response: Correct the configuration file and restart the server.

IMW0004I gc-mem-usage must be >20. Using 100. Small values prevent gc from working efficiently. You suggested: *specifier.*

Explanation: You entered a *specifier* that is not valid or not within range.

User Response: Correct the configuration file.

IMW0005E Insufficient parameters for directive: *directive.*

Explanation: This directive requires more parameters than provided.

User Response: Correct the configuration file.

IMW0006E Must be MVS UserID, not UID: *number***.**

Explanation: You entered a numeric UID, not a surrogate MVS user ID. The same UID number could be assigned to several MVS user IDs. To avoid ambiguity when accessing MVS resources, the server requires a surrogate MVS user ID. The UID and GID used for accessing z/OS UNIX System Services resources is assigned from the specified surrogate MVS user ID.

User Response: Specify a surrogate MVS user ID or %%CLIENT%%.

IMW0007E Syntax error in configuration file. Expecting '{' to start protection definition of: *name***.**

Explanation: The protection setup in the configuration file is not valid.

The correct syntax is:

```
Protection name {
protection directive
⋮
}
```

User Response: Correct the configuration file.

IMW0008E **The cache lock timeout specifier is not valid:** *time specifier.*

Explanation: You entered a *time specifier* that is not a valid amount of time.

User Response: Correct the configuration file.

IMW0009E **The cache time margin specifier is not valid:** *time specifier.*

Explanation: You entered a *time specifier* that is not a valid amount of time.

User Response: Correct the configuration file.

IMW0010E **The cache_clean_def time specifier is not valid:** *time specifier.*

Explanation: You entered a *time specifier* that is not a valid amount of time.

User Response: Correct the configuration file.

IMW0011E **The cache_unused_def time specifier is not valid:** *time specifier.*

Explanation: You entered a *time specifier* that is not a valid amount of time.

User Response: Correct the configuration file.

IMW0012E **The CacheLastModifiedFactor is not valid:** *string.*

Explanation: You entered a *string* that the server could not interpret as a real number (for example, 0.25) or a negative string.

User Response: Correct the configuration file.

IMW0013E **The configuration directive is not valid:** *directive.*

Explanation: You entered a *directive* that is not valid or not spelled correctly.

User Response: Correct the configuration file.

IMW0014E **The daily gc time specifier is not valid:** *time specifier.*

Explanation: You entered a *time specifier* that is not valid or a negative string.

User Response: Correct the configuration file.

IMW0015E **The default expiry time specifier is not valid:** *time specifier.*

Explanation: You entered a *time specifier* that is not valid.

User Response: Correct the configuration file.

IMW0016E **The Dir directive is not recognized:** *directive.*

Explanation: You entered a *directive* starting with Dir that is not valid or not spelled correctly.

User Response: Correct the configuration file.

IMW0017E **The DirAccess mode is not recognized:** *mode.*

Explanation: You entered a *mode* that is not valid or not spelled correctly. The mode can be `selective`, a positive string, or a negative string.

User Response: Correct the configuration file.

IMW0018E **The DirReadme mode is not recognized:** *mode.*

Explanation: You entered a *mode* that is not valid or not spelled correctly. The mode can be `top`, `bottom`, or a negative string.

User Response: Correct the configuration file.

IMW0019E **The DirShow directive is not valid:** *directive.*

Explanation: You entered a *directive* that is not valid or not spelled correctly.

User Response: Correct the configuration file.

IMW0020E **The DirShowMaxDescriptionLength parameter is not valid:** *parameter.*

Explanation: You entered a *parameter* which the server could not interpret as an integer.

User Response: Correct the configuration file.

IMW0021E **The DirShowMaxLength parameter is not valid:** *parameter.*

Explanation: You entered a *parameter* which the server could not interpret as an integer.

User Response: Correct the configuration file.

IMW0022E **The DirShowMinLength parameter is not valid:** *parameter.*

Explanation: You entered a *parameter* that the server could not interpret as an integer.

User Response: Correct the configuration file.

IMW0023E **The FTPDirInfo parameter is not valid:** *parameter.*

Explanation: You entered a *parameter* that is not valid or not spelled correctly. The parameter can be `top`, `bottom`, or a negative string.

User Response: Correct the configuration file.

IMW0024E **The Icon directive is not recognized:** *directive.*

Explanation: You entered a *directive* that is not valid or not spelled correctly.

User Response: Correct the configuration file.

IMW0025E **The logfile format is not recognized (use Old or Common):** *format.*

Explanation: You entered a *format* that is not valid or not spelled correctly.

User Response: Correct the configuration file.

IMW0026E **The logtime is not recognized (use GMT or LocalTime):** *parameter.*

Explanation: You entered a *parameter* that is not valid or not spelled correctly.

User Response: Correct the configuration file.

IMW0027E **The method you tried to disable is not valid:** *method name.*

Explanation: You entered a *method* that is not valid or not spelled correctly.

User Response: Correct the configuration file.

IMW0028E **The method you tried to enable is not valid:** *method name.*

Explanation: You entered a *method* that is not valid or not spelled correctly.

User Response: Correct the configuration file.

IMW0029E **The number of parameters to directive is not valid:** *directive.*

Explanation: You entered either too few or too many parameters for this directive.

User Response: Correct the configuration file.

IMW0030E **The parameter to Port directive is not valid:** *parameter.*

Explanation: You entered a *parameter* that the server could not interpret as an integer.

User Response: Correct the configuration file.

IMW0031E **The protection setup name is not defined:** *name.*

Explanation: You entered a *name* which is either not defined or not correctly spelled.

User Response: Correct the configuration file.

IMW0032E **The proxy directive is not recognized:** *directive.*

Explanation: You entered a *directive* which is either not defined or not correctly spelled.

User Response: Correct the configuration file.

IMW0033E **The SecurityLevel parameter is not valid:** *parameter.*

Explanation: You entered a *parameter* that is not valid. You can enter either `normal` or `high`.

User Response: Correct the configuration file.

IMW0034E **The ServerType is not recognized:** *string.*

Explanation: You entered a *string* that is not valid or not spelled correctly. The Web server supports only ServerType StandAlone.

User Response: Correct the configuration file.

IMW0035E **The timeout specifier is not valid:** *time specifier.*

Explanation: You entered a *time specifier* that is not valid.

User Response: Correct the configuration file.

IMW0036E **There are too many arguments on one line in the configuration file. Max:** *number.*

Explanation: You entered more arguments than are supported for any configuration directive.

User Response: Correct the configuration file.

IMW0037E **Too many parameters for CacheLastModifiedFactor:** *number.*

Explanation: You entered *number* strings on this line. CacheLastModifiedFactor only accepts one parameter, a real number.

User Response: Correct the configuration file.

IMW0038E Cannot open the cache access log: *name.*

Explanation: The server was unable to open the log file. Verify the log file name, the permissions, the user mask, and the available space in the file system.

User Response: Correct the log file name specified in the configuration file or correct the file system problem.

IMW0039E Cannot open the error log: *name.*

Explanation: The server was unable to open the log file. Verify the log file name, the permissions, the user mask, and the available space in the file system.

User Response: Correct the log file name specified in the configuration file or correct the file system problem.

IMW0040E Cannot open the log file: *name.*

Explanation: The server was unable to open the log file. Verify the log file name, the permissions, the user mask, and the available space in the file system.

User Response: Correct the log file name specified in the configuration file or correct the file system problem.

IMW0041E Cannot open the proxy access log: *name.*

Explanation: The server was unable to open the log file. Verify the log file name, the permissions, the user mask, and the available space in the file system.

User Response: Correct the log file name specified in the configuration file or correct the file system problem.

IMW0042E Cannot open the trace log. *name.*

Explanation: The server was unable to open the log file. Verify the log file name, the permissions, the user mask, and the available space in the file system.

User Response: Correct the log file name specified in the configuration file or correct the file system problem.

IMW0043E Relative log name when ServerRoot not specified: *directive.*

Explanation: Log file names that do not begin with a / are relative to the ServerRoot directory.

User Response: Specify a ServerRoot directory or use absolute log names.

IMW0044E There were more errors, but queued only: *number.*

Explanation: Errors that occur prior to opening the error log are queued in memory. More errors occurred than could be queued.

User Response: Correct the indicated errors and restart the server.

IMW0045S Could not create adult_mtx.

Explanation: The server was not able to create a required mutex (lock) at initialization.

User Response: Verify that the Web server has sufficient virtual memory and is running with the required level of z/OS UNIX System Services.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0046S Could not create auth_mtx.

Explanation: The server was not able to create a required mutex (lock) at initialization.

User Response: Verify that the Web server has sufficient virtual memory and is running with the required level of z/OS UNIX System Services.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0047S Could not create free_presentation_mtx.

Explanation: The server was not able to create a required mutex (lock) at initialization.

User Response: Verify that the Web server has sufficient virtual memory and is running with the required level of z/OS UNIX System Services.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0048S Could not initialize the translate tables (iconv).

Explanation: The server was not able to open the iconv translation service for translation between ASCII ISO8859-1 and EBCDIC IBM-1047.

User Response: Verify that the Web server is running with the required level of z/OS UNIX System Services.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0049E Cannot open protection setup file: *name*.

Explanation: A Protect directive referenced a protection setup file that the server was unable to open for reading.

User Response: Verify that the file name is spelled correctly in the configuration file. Verify that the file exists and has the correct file permissions. Correct the

configuration file or protection setup file.

IMW0050E Group not found: *name.*

Explanation: The group *name* is not defined or is not spelled correctly.

User Response: Verify the name in your protection mask directives and group file. Correct the configuration, protection setup, group or .acl file.

IMW0051E User not found: *name.*

Explanation: The user *name* is not defined or is not spelled correctly.

User Response: Verify the name in your protection mask directives and group file. Correct the configuration, protection setup, group or .acl file.

IMW0052E You did not specify a protection file in the Protect directory. Default protection was not set.

Explanation: The Protect directive that was matched for this request did not have an in-line protection setup, a named protection setup or a protection setup file name. This is only allowed when there is a preceding DefProt directive that also matches the current request. The client's request is refused.

User Response: Correct the configuration file.

IMW0053E You did not specify the required protection file in the DefProt directive.

Explanation: The DefProt directive must always have an in-line protection setup, a named protection setup or a protection setup file name.

User Response: Correct the configuration file.

IMW0054E Cannot fstat() the new cache file: *name***.**

Explanation: This is either an internal logic error or an HFS system problem.

User Response:

- Verify the cache root name defined in the configuration file.
- Verify that the server has appropriate permissions to the cache root and that the HFS is not damaged.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0055E Cannot get the cache information for: *name***.**

Explanation: This is either an internal logic error or HFS system problem.

User Response:

- Verify the cache root name defined in the configuration file.
- Verify that the server has appropriate permissions to the cache root and that the HFS is not damaged.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0056E Cannot write the cache information entry for: *string***.**

Explanation: This is either an internal logic error or HFS system problem.

User Response:

- Verify the cache root name defined in the configuration file.
- Verify that the server has appropriate permissions to the cache root and that the HFS is not damaged.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0057E Cannot write the cache information entry for: *name***.**

Explanation: This is either an internal logic error or HFS system problem.

User Response:

- Verify the cache root name defined in the configuration file.
- Verify that the server has appropriate permissions to the cache root and that the HFS is not damaged.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0058E Content-Length mismatch when retrieved: *name***.**

Explanation: This is either an internal logic error or HFS system problem.

User Response:

- Verify the cache root name defined in the configuration file.
- Verify that the server has appropriate permissions to the cache root and that the HFS is not damaged.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0059E HTLoadCacheToStream read error. Read returns: *number***.**

Explanation: This is either an internal logic error or HFS system problem.

User Response:

- Verify the cache root name defined in the configuration file.
- Verify that the server has appropriate permissions to the cache root and that the HFS is not damaged.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0060E The ExecDirPass mode is not valid: *mode.*

Explanation: You entered a *mode* that is not valid. The mode can be a positive or negative string.

User Response: Enter a positive or negative value for ExecDirPass in the configuration file.

IMW0062I CacheTrace.. On

Explanation: You have requested cache access tracing in your configuration.

User Response: None.

IMW0063E Cannot bind and listen on port *number.* **Ensure TCP/IP is configured and running and that httpd is not already running.**

Explanation: The server was not able to get ownership of the specified TCP port.

User Response: Make the port available for use. Verify that:

- There must be an AF_INET transport provider connection to z/OS UNIX System Services.
- The port number is reserved for use by z/OS UNIX in all connected AF_INET transport providers.
- If a HostName is specified, there must not be any other application already using the specified port on the same IP address or on all z/OS UNIX IP addresses.

IMW0064E Cannot create socket.

Explanation: Indicates a problem with TCP/IP.

User Response: Make sure TCP/IP is up and running.

IMW0065E Cannot create socket.

Explanation: This is an internal error.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0066E Cannot get passwd entry for uid: *number.*

Explanation: This is an internal error.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0067E Cannot get passwd entry for user *string*

Explanation: This is an internal error.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0068E Cannot initialize groups for user: *id.*

Explanation: This is an internal error.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0069E Cannot initialize groups for user *name.*

Explanation: This is an internal error.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0070E Cannot open cache file for reading: *name.*

Explanation: This is either an internal logic error or HFS system problem.

User Response: Verify:

- The cache root name defined in the configuration file.
- The server has appropriate permissions to the cache root and that the HFS is not damaged.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0071E Cannot open gc report file for reading: *name***.**

Explanation: The garbage collection (cache cleanup) thread failed to complete normally.

User Response: Check the error and trace logs for correctable problems.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0072E Cannot open pid file for writing: *name.*

Explanation: The pid file cannot be opened.

User Response: Verify the following:

- The name specified in the configuration file is valid.
- The file system is read/write.
- The file permissions are correct in the path.
- The server has the appropriate permissions.

Correct any problems found with the pid file or configuration.

IMW0073E Cannot open pid file: *name.*

Explanation: The pid file cannot be opened.

User Response: Verify the following:

- The name specified in the configuration is valid.
- The file has not been erased or damaged by another user.
- The file permissions are correct in the path.
- The server has the appropriate permissions.

Correct any problems found with the pid file or configuration.

IMW0074E Cannot read pid from *name***.**

Explanation: The pid file cannot be read.

User Response: Verify the following:

- The name specified in the configuration is valid.
- Verify that the file has not been erased or damaged by another user.
- Verify that the file permissions are correct in the path.
- Verify that the server has the appropriate permissions.

Correct any problems found with the pid file or configuration.

IMW0075E Cannot set group id to: *number.*

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0076E Cannot set parent group id to: *number***.**

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0077E Cannot set parent user id to: *number***.**

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0078E Cannot set socket option.

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0079E Cannot set socket option.

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0080E Cannot set user id to: *number.*

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0081E Configuration file is required.

Explanation: Indicates configuration file could not be found.

User Response: Provide a valid configuration file.

IMW0082E Connection interrupted (SIGPIPE)

Explanation: The client broke the connection prior to sending all of the request.

User Response: None.

IMW0083E Connection interrupted (SIGPIPE), req: *request*

Explanation: The client broke the connection prior to receiving all of the data.

User Response: None.

IMW0084E CRASHED (bus error, core dumped).

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0085E ABEND signal *signal_number* **has been caught for thread type** *thread_number*

Explanation: The Web server's ABEND recovery signal handler has detected an ABEND condition. A CEEDUMP or both a CEEDUMP and an IEATDUMP may be taken for this error. The Web server may recover from this error.

signal_number can be:

4 SIGILL (Illegal instruction): An attempt was made to execute an illegal instruction.

8 SIGFPE (Fixed-point exception): An attempt was made to perform an arithmetic operation that would have resulted in an overflow condition, such as a divide by zero.

11 SIGSEGV (Segmentation violation): An attempt was made to access inaccessible storage.

thread_number can be:

0 Accept thread
1 Worker thread
2 Log thread
3 Garbage collection thread
4 Restart thread
5 Timer thread
6 SNMP thread
7 Network monitor thread
8 Performance monitor thread
9 JVM thread
10 JVM processor thread
11 Inter-process communication thread
12 FRCA log thread (for the Fast Response Cache Accelerator)

User Response: If a CEEDUMP was taken or both a CEEDUMP and an IEATDUMP were taken, determine if the ABEND occurred in Web server code or in a plug-in or other code. If the ABEND occurred in Web server code, report the problem to the IBM Software Support Center. If the Web server recovered from the ABEND, stop and restart the Web server as soon as you can to ensure that resources are properly cleaned up.

IMW0086E Crashing request was: *string.*

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0087E Extra command line parameter: *parameter.*

Explanation: Additional unexpected parameters were specified on the // EXEC JCL statement.

User Response: Correct the JCL and resubmit.

IMW0088E Extra parameter: *parameter.*

Explanation: Additional unexpected parameters were specified on the // EXEC JCL statement.

User Response: Correct the JCL and resubmit.

IMW0089E Forbidden by rule.

Explanation: This string is returned by the server in the HTTP status line on requests that are not allowed by your configuration file.

User Response: None.

IMW0090E gc report file is empty: *string.*

Explanation: The garbage collection (cache cleanup) thread failed to complete normally.

User Response: Check the error and trace logs for correctable problems.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0091E gc report has wrong process id. Real gc might have crashed.

Explanation: The garbage collection (cache cleanup) thread failed to complete normally.

User Response: Check the error and trace logs for correctable problems.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0092E gc was returned, but caching is not occurring.

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0093E Method *method name.* **is not enabled on this server.**

Explanation: This string is returned by the server in the HTTP status line on requests that specify a method name that is not enabled in your configuration.

User Response: For this request to succeed, explicitly enable this method in your configuration.

IMW0094E Method *method name.* **is not valid or not implemented.**

Explanation: This string is returned by the server in the HTTP status line on requests that specify an unrecognized method name.

User Response: None.

IMW0095E Permission denied.

Explanation: This request failed. This string is returned by the server in the HTTP status line of the request.

User Response: Verify that the user ID that the server is running under has the appropriate permissions.

IMW0096E Request parsing failed.

Explanation: The server cannot complete a request. Possible causes are:

- The client cancelled the request before it completed.
- A TCP/IP problem occurred.
- The server forced a reconnection because a server timeout occurred or PersistTimeout was set to 0.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0097E Restart failed.

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0098I Restart succeeded.

Explanation: The restart signal was successfully sent to the server.

User Response: None.

IMW0099I Restarting.. httpd

Explanation: The server has quiesced all requests and is reloading the configuration file.

User Response: None.

IMW0100E Script timed out.

Explanation: A CGI program has failed to complete processing within the time specified in your configuration and will be terminated.

User Response: None.

IMW0101I Sending..... HUP signal to process: *number.*

Explanation: A warning signal has been sent to a CGI program that has timed out.

User Response: None.

IMW0102E SETUID ERROR: there is no such user: *string.*

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0103I SIGHUP caught: reloading configuration files.

Explanation: The server has received the restart signal.

User Response: None.

IMW0104E The -d option is not valid: *string*

Explanation: The -dxx parameter (diraccess and dirreadme override) is not specified correctly.

User Response: Correct the JCL and resubmit.

IMW0105E The -r option was specified, so the dir parameter was ignored: *string.*

Explanation: Rule (configuration) file name must be specified with -r.

User Response: None.

IMW0106E The cache root was not specified when started in gc-only mode

Explanation: A cache root is required to run garbage collection.

User Response: Correct the configuration.

IMW0107E The cache root was not specified when started in gc-only mode *string*

Explanation: A cache root is required to run garbage collection.

User Response: Correct the configuration.

IMW0108E The command line option is not recognized: *string.*

Explanation: You entered an *option* that is either not defined or not correctly spelled.

User Response: Correct the JCL and resubmit.

IMW0109E The command line option is not recognized: *string.*

Explanation: You entered an *option* that is either not defined or not correctly spelled.

User Response: Correct the JCL and resubmit.

IMW0110E The parameter to -p option is not valid: *parameter.*

Explanation: You entered a *parameter* that is not a valid port number.

User Response: Correct the JCL and resubmit.

IMW0111E The request is not valid: *string.*

Explanation: The server received an HTTPD request string that is not properly formatted. This may be from an improperly written client or a user trying to break in by telneting to the server port.

User Response: None.

IMW0112E The request is not valid or not recognized: *string.*

Explanation: The server received an HTTPD request string that contained an unrecognized method.

User Response: None.

IMW0113E There is no parameter for -errlog option.

Explanation: This parameter must be followed by the errorlog name.

User Response: Correct the JCL and resubmit.

IMW0114E There is no parameter for -errlog option.

Explanation: This parameter must be followed by the errorlog name.

User Response: Correct the JCL and resubmit.

IMW0115E There is no such process.

Explanation: The server was started with the -restart parameter. The pid file specified in the configuration contains an incorrect PID number. Either the server is no longer running or the wrong pid file was used.

User Response: Correct the JCL and resubmit.

IMW0116E Timeout occurred when reading request.

Explanation: A complete request was not received from the client within the amount of time specified in the configuration. The connection from the client has been closed.

User Response: None.

IMW0117E Timeout occurred when sending response.

Explanation: The server was not able to complete sending the response within the amount of time specified in the configuration. The connection from the client has been closed.

User Response: None.

IMW0118E Translated NULL when should do search.

Explanation: A search request URL was received from the client that did not contain a QUERY_STRING.

User Response: None.

IMW0119E Unknown error occurred.

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0120E You did not specify a parameter for -l option.

Explanation: This parameter must be followed by the name of the log.

User Response: Correct the JCL and resubmit.

IMW0121E You did not specify a parameter for -l/-newlog/-oldlog option

Explanation: This parameter must be followed by the name of the log.

User Response: Correct the JCL and resubmit.

IMW0122E Empty item not allowed.

Explanation: Your group file is missing field specific information.

User Response: Correct the group file.

IMW0123E Expected address part (single address or list).

Explanation: Item in message is missing from the group file.

User Response: Correct the group file.

IMW0124E Expecting ')' closing address list.

Explanation: Item in message is missing from the group file.

User Response: Correct the group file.

IMW0125E Expecting ')' closing user/group list.

Explanation: Item in message is missing from the group file.

User Response: Correct the group file.

IMW0126E Expecting a single address or '(' beginning list.

Explanation: Item in message is missing from the group file.

User Response: Correct the group file.

IMW0127E Expecting a single name or '(' beginning list.

Explanation: Item in message is missing from the group file.

User Response: Correct the group file.

IMW0128E Expecting an address template.

Explanation: Item in message is missing from the group file.

User Response: Correct the group file.

IMW0129E Expecting field separator.

Explanation: Item in message is missing from the group file.

User Response: Correct the group file.

IMW0130E Expecting group name.

Explanation: Item in message is missing from the group file.

User Response: Correct the group file.

IMW0131E Expecting user or group name.

Explanation: Item in message is missing from the group file.

User Response: Correct the group file.

IMW0132I Group is NULL

Explanation: The group has no name.

User Response: None.

IMW0133I Group: *group_name.*

Explanation: Indicates group name.

User Response: None.

IMW0134I NULL-ITEM

Explanation: Indicates an empty item in group file.

User Response: None.

IMW0135I NULL RECORD

Explanation: Indicates an empty line in the group file.

User Response: None.

IMW0136E There is garbage after group definition.

Explanation: There is unrecognized information in the group file.

User Response: None.

IMW0137I Accepted

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0138E An error occurred that has no explanation. Call IBM software support.

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0139I Created

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0140E Forbidden

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0141I Found

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0142E Internal error. This is a software problem. Call IBM software support.

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0143E Method

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0144E Moved

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0145E No response

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0146E Not authorized.

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0147E Not found.

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0148E Not implemented

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0149E Not modified

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0150I OK

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0151E Partial information

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0152E Payment required.

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0153E The request is not valid.

Explanation: Returned to client program in the HTTP status line.

User Response: None.

IMW0154E Cannot open cache report file for writing: *string.*

Explanation: Verify the file name and permissions. Verify that the file is system is mounted, read/write and not full.

User Response: Correct any problems found.

IMW0155E Cannot rewrite cache info file in directory: *string.*

Explanation: Verify the file name and permissions. Verify that the file is system is mounted, read/write and not full.

User Response: Correct any problems found.

IMW0156E Internal error: spawn() failed. errno=*return_code***, errno2=***reason_code***, errmsg:** *string*

Explanation: The server received an error when it tried to spawn a new address space to run a CGI program. The z/OS UNIX System Services error information and message are displayed.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Verify:
- That the file permissions allow execution under the user ID used to process this request.

- You have not exceeded the number of z/OS UNIX address spaces allowed.

Correct the z/OS UNIX environment problem.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0157E Internal error: Cannot create pipe.

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0158E Internal error: Cannot read script output pipe.

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0159E The preparse script request is not valid. *string* is not executable.

Explanation: The preparse script program *string* is not marked as executable. Verify that the HTML page, the script name and permissions are correct.

User Response: Correct any problems.

IMW0160E The script execution request is not valid.

Explanation: The script program is not found. Verify that the HTML page, the script name and permissions are correct.

User Response: Correct any problems.

IMW0161E The script request is not valid. No variation of *string* is executable.

Explanation: An executable script program is not found. The server also looked for variations of the named program by appending .sh and .pp to the requested name.

If the following message is accompanied by Error 500, this indicates that the helpout file cannot be executed because of a permission bit problem.

```
IMW0161E The script request is not valid. No
variation of
/usr/lpp/internet/server_root/admin-bin/helpout
is executable.
```

User Response: Verify that the HTML page, the script name and permissions are correct. Correct any problems.

If you suspect a permission bit problem, check permission bits along the directory structure using the **ls -dl** command to make sure that the execute bit is turned on. The following example shows a valid set of permission bits:

```
$ ls -dl /
drwxr-xr-x 37 IBMUSER OMVSGRP 0 Sep 29 09:01 /
$ ls -dl /usr
drwxr-xr-x 14 IBMUSER OMVSGRP 0 Jul 15 13:52 /usr
$ ls -dl /usr/lpp
drwxr-xr-x 18 IBMUSER OMVSGRP 0
   Aug 20 07:27 /usr/lpp
$ ls -dl /usr/lpp/internet
drwxr-xr-x  9 USER88 WEBTEAM 0 Aug 20 14:37
/usr/lpp/internet
$ ls -dl /usr/lpp/internet/server_root
drwxr-xr-x 14 USER88 WEBTEAM 0 Sep 28 13:13
/usr/lpp/internet/server_root
$ ls -dl /usr/lpp/internet/server_root/admin-bin
drwxr-xr-x  4 USER88 WEBTEAM 0 Aug 20 14:20
/usr/lpp/internet/server_root/admin-bin
$ ls -dl /usr/lpp/internet/server_root
   /admin-bin/helpout
-rwxr-xr-x  1 USER88 WEBTEAM 704512 Aug 12 17:10
/usr/lpp/internet/server_root/admin-bin/helpout
```

Note: In this example some commands or responses appear on two lines for printing purposes.

IMW0162E Internal error running function *function_name* from DLL module *file_name*

Explanation: This message is sent to a client when an ABEND occurs in a plug-in running on behalf of a request sent by the client. The plug-in by the name of *function_name* from DLL *file_name* was executed by the Web server. The Web server was able to recover from the ABEND. A CEEDUMP or both a CEEDUMP and an IEATDUMP may have been taken. Message IMW0085E should also appear in the error log.

User Response: Retry the request. If the failure occurs again, correct the error before retrying. See message IMW0085E for additional information.

IMW0163E RFC931: bind() failed: BIND_ERROR

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0164E RFC931: fdopen() failed: FDOPEN_ERROR

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0165E RFC931: socket() failed: SOCKET_ERROR

Explanation: This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0166E An error occurred while allocating storage for SSI processing

Explanation: During processing of a server-side include request, a memory allocation request failed.

User Response: Ensure there is enough memory for the server.

IMW0167E *directive* **failed: var =** *variable_text*, **value** *value_text*

Explanation: During processing of a server-side include, the server attempted to resolve *directive* but failed. The *variable_text* can be specified using: VAR, FILE, CMNTMSG, ERRORMSG, TIMEFMT, or SIZEFMT.

User Response: Correct the error.

See "Using server-side includes to insert information into CGI programs and HTML documents" on page 214 for more information on using server-side includes and directives.

IMW0168E Expected *name* **but found** *string*

Explanation: During processing of a server-side include or configuration directive, the server encountered *string* when it was expecting *name*.

User Response: Correct the error.

See "Using server-side includes to insert information into CGI programs and HTML documents" on page 214 for more information on using server-side includes and directives.

For more information on configuration directives, see Appendix B, "Configuration directives" on page 417.

IMW0169E Internal server error: Imbedded BINARY code page is not supported.

Explanation: During processing of a server-side include request, the server attempted to process binary data when it is already processing non-binary data.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0170E Stat on file failed: *name*

Explanation: During processing of a server-side include request, the server attempted to gather statistics on file *name*, but failed.

User Response: Verify *name* exists and the permissions are correct.

IMW0171E strftime() call failed: TIMEFMT *timefmt*, **file** *name*, **errno=***return_code*, **errno2=***reason_code*.

Explanation: During processing of a server-side include request, the server attempted a call to the C function strftime() with the time format string *timefmt* for document *name*.

User Response: Verify the time format string is correct and the server has enough memory to process the request.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

See "Using server-side includes to insert information into CGI programs and HTML documents" on page 214 for more information on using server-side includes and directives.

IMW0172E The directive was not complete on HTSSI_abort

Explanation: During processing of a server-side include request, abnormal termination of the output stream occurred while processing a server-side include directive.

User Response: Correct the error.

See "Using server-side includes to insert information into CGI programs and HTML documents" on page 214 for more information on using server-side includes and directives.

IMW0173E The directive was not complete on HTSSI_free

Explanation: During processing of a server-side include request, termination of the output stream occurred while processing a server-side include directive.

User Response: Correct the error.

See "Using server-side includes to insert information into CGI programs and HTML documents" on page 214 for more information on using server-side includes and directives.

IMW0174E The directive is not supported: *directive*

Explanation: During processing of a server-side include request, the document contained a *directive* which is not valid or not spelled correctly.

User Response: Correct the error.

See "Using server-side includes to insert information into CGI programs and HTML documents" on page 214 for more information on using server-side includes and directives.

IMW0175E The directive was not valid: *string*

Explanation: During processing of a server-side include request, the document contained a directive that was not valid or not correctly spelled.

User Response: Correct the error.

See "Using server-side includes to insert information into CGI programs and HTML documents" on page 214 for more information on using server-side includes and directives.

IMW0176E The text was not valid: *string***:**

Explanation: During processing of a server side include request, the directive text *string* was found to contain an illegal character.

User Response: Correct the error.

See "Using server-side includes to insert information into CGI programs and HTML documents" on page 214 for more information on using server-side includes and directives.

IMW0177E SIZEFMT is not valid: *value*

Explanation: During processing of a server-side include request, the document contained a file size format *value* which is either not valid or not correctly spelled. Valid values are **bytes** and **abbrev**.

User Response: Correct the error.

See "Using server-side includes to insert information into CGI programs and HTML documents" on page 214 for more information on using server-side includes and directives.

IMW0178E There was an illegal attempt to use .. in file name *name.*

Explanation: During processing of a server-side include request, the document attempted to access the document *name* which is outside SSI_ROOT and its descendents.

User Response: Correct the error.

See "Using server-side includes to insert information into CGI programs and HTML documents" on page 214 for more information on using server-side includes and directives.

IMW0179E VARREF was not valid: varref

Explanation: During processing of a server-side include request, the variable reference *varref* was found to contain an illegal character.

User Response: Correct the error.

See "Using server-side includes to insert information into CGI programs and HTML documents" on page 214 for more information on using server-side includes and directives.

IMW0180E Not authorized. Proxy-Authentication failed (or your browser does not support it).

Explanation: The proxy request requires a user ID and password.

User Response: Provide a user ID and password if the browser supports the PROXY-AUTHENTICATE function.

IMW0181E The Idle thread timeout specifier is not valid: *timeout specifier*

Explanation: You entered a timeout specifier that is not valid.

User Response: Correct the configuration file.

IMW0182E The HomeDirs directive is not valid. You did not specify a directory.

Explanation: You entered a HomeDirs directive that is not valid.

User Response: Correct the configuration file.

IMW0183E Unknown error type *error type*

Explanation: The ErrorPage directive in the configuration file is not correct.

User Response: Correct the configuration file.

IMW0184E No error file specified for error type *error type*

Explanation: The ErrorPage directive in the configuration file is not correct.

User Response: Correct the configuration file.

IMW0185E The AccessLogExcludeURL directive is not valid. You did not specify a Method.

Explanation: You entered an AccessLogExcludeURL directive that is not valid.

User Response: Correct the configuration file.

IMW0186E The AccessLogExcludeMethod directive is not valid. You did not specify a Method.

Explanation: You entered an AccessLogExcludeMethod directive that is not valid.

User Response: Correct the configuration file.

IMW0187E The AccessLogExcludeReturnCode directive is not valid. You did not specify a Return Code.

Explanation: You entered an AccessLogExcludeReturnCode directive that is not valid or did not enter a Return Code.

User Response: Correct the configuration file.

IMW0188E The AccessLogExcludeMimeType directive is not valid. You did not specify a MimeType.

Explanation: You entered an AccessLogExcludeMimeType directive that is not valid.

User Response: Correct the configuration file.

IMW0189E Invalid API directive.

Explanation: You entered an API directive that is not valid.

User Response: Correct the configuration file.

IMW0190E Invalid parameter for UseAcls directive: *parameter*

Explanation: You entered a parameter for the UseAcls directive that is not valid.

User Response: Correct the configuration file.

IMW0191E iconv() does not support the combination of codepages specified.

Explanation: Specifications for DefaultFsCp and DefaultNetCp in the configuration file are not compatible.

User Response: Verify and correct the code pages that the server runs under.

IMW0192E Cannot open the CGI error log: *name*

Explanation: The server was unable to open the CGI log file. Verify the log file name, the permissions, the user mask, and the available space in the file system.

User Response: Correct the log file name specified in the configuration file or correct the file system problem.

IMW0193I OK

Explanation: The page was served successfully.

User Response: None.

IMW0194I OK-GATEWAY

Explanation: The page was served successfully.

User Response: None.

IMW0195I OK-REDIRECT

Explanation: The page was served successfully.

User Response: None.

IMW0196I NOT AUTHENTICATED

Explanation: The user was not authenticated, so the page was not served.

User Response: Enter a valid user ID and password.

IMW0197E NOT AUTHORIZED

Explanation: The user was not in the mask.

User Response: Enter a user that is in the ACL for the page.

IMW0198E FORBIDDEN BY IP

Explanation: The IP address of the browser is not in the group.

User Response: Add the IP address to the group.

IMW0199E FORBIDDEN BY IP FOR PROXY

Explanation: Running as a proxy server.

User Response: Add the IP address to the group.

IMW0200E FORBIDDEN BY RULE

Explanation: Could not get a page because it did not pass a rule.

User Response: Correct the rule.

IMW0201E NO ACL FILE

Explanation: Could not read any of the ACL files.

User Response: Create ACL files, or set ACL settings to NEVER.

IMW0202E NO ACL ENTRY

Explanation: There are no allowed groups in the Access Control List (ACL) file.

User Response: Include groups in the ACL file or turn checking off.

IMW0203E * SETUP ERROR *

Explanation: Protection file not found or syntax error in file.

User Response: Correct the file.

IMW0204E ..IN URL

Explanation: Cannot have .. in URL.

User Response: Remove .. from URL.

IMW0205E HTBIN OFF

Explanation: htbin is not enabled on your server.

User Response: Enable htbin on your server.

IMW0206E INVALID REDIRECT

Explanation: The configuration file contains a redirect that is not valid.

User Response: Correct the redirection.

IMW0207E NO SUCH USER

Explanation: The user directory is not correct.

User Response: Correct the configuration file.

IMW0208E PUT NOT ALLOWED

Explanation: PUT/DELETE must be explicitly allowed.

User Response: Correct the configuration file.

IMW0209E NOT FOUND

Explanation: The file does not exist or is read protected.

User Response: Correct the configuration file.

IMW0210E MULTI FAILED

Explanation: The file does not exist or is read protected.

User Response: Correct the configuration file.

IMW0211E Contact IBM Service

Explanation: An internal error occurred.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0212E Cannot find help file *name*. Help is disabled.

Explanation: The help file you requested cannot be found.

User Response: Try again.

IMW0213E Cannot initialize application Help. Help is disabled.

Explanation: The help application cannot be initialized.

User Response: Correct the configuration.

IMW0214E Help is not available.

Explanation: Help is not available.

User Response: None.

IMW0215E Help is not available for this item.

Explanation: Help is not available for the information.

User Response: None.

IMW0216E Not authorized. Authentication failed.

Explanation: You are not allowed to do the function you are trying to do.

User Response: Verify the user ID, password, and protect rule.

IMW0217E Not authorized to access the document.

Explanation: You are not allowed to access the document.

User Response: Contact your system programmer to increase your privilege.

IMW0218E Forbidden by rule.

Explanation: You specified an action that is forbidden by the rules specified in the configuration file.

User Response: Correct the configuration file.

IMW0219E Server will not serve to your IP address.

Explanation: The mask specifications indicate that the client is not coming from a supported IP address.

User Response: Correct the configuration file.

IMW0220E Proxy server will not serve to your IP address (at least with this HTTP method).

Explanation: The mask specifications indicate that the client is not coming from a supported IP address.

User Response: Correct the configuration file.

IMW0221E Access to this file is not allowed (no ACL file).

Explanation: The file is protected, but no mask or ACL is specified. This is mandatory for proxy protection.

User Response: Correct the configuration file.

IMW0222E Access to this file is not allowed (no ACL entry).

Explanation: No ACL entry is specified.

User Response: Correct the configuration file.

IMW0223E Server protection setup error occurred. Probably the protection setup file was not found or it contained a syntax error.

Explanation: The protection setup file was not found or the file contains a syntax error.

User Response: Verify the setup file exists or correct the syntax error.

IMW0224E Forbidden - URL containing ... forbidden (do not try to break in)

Explanation: The URL specified is not valid. You are not allowed to specify a period (`.`) in the URL.

User Response: Try again.

IMW0226E The redirection in the configuration file is not valid.

Explanation: There is no destination for the specified redirect.

User Response: Verify the configuration.

IMW0227E The user directory is not valid.

Explanation: The user directory is not valid.

User Response: Verify the UserDir directive in the configuration file.

IMW0228E The PUT and DELETE methods must be specified in the server's protection setup.

Explanation: Methods must be explicitly specified or allowed.

User Response: Correct your configuration file.

IMW0229E The file was not found, even after searching on any extensions to the file name *filename*. The file does not exist or is read-protected.

Explanation: The file *filename* was not found.

User Response: Verify existence of the file or change its protection.

IMW0230I Document follows.

Explanation: The server is indicating the requested document follows.

User Response: None.

IMW0231I Gatewaying.

Explanation: The server is acting as a gateway.

User Response: None.

IMW0232I Found

Explanation: The redirection specified is valid.

User Response: None.

IMW0233E Access denied. Cannot specify reason. Call IBM software support.

Explanation: Server access is not allowed.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0234I Starting httpd

Explanation: Server initialization is in progress.

User Response: None.

IMW0235I Server is ready.

Explanation: Server initialization is complete. The server is ready to serve client requests.

User Response: None.

IMW0236E Access denied - password expired. Enter *old_password/new_password/new_password* **to change your password.**

Explanation: Your password has expired.

User Response: Enter your old password and your new password twice.

IMW0237E Password changed. Enter *newpw* **to continue.**

Explanation: Your password has been changed corrected.

User Response: None.

IMW0238E New passwords are not equal, try again. Enter *old_password/new_password/new_password* **to change your password.**

Explanation: The two new passwords you entered are not the same.

User Response: The two new passwords must be the same. Try again.

IMW0239E New password has invalid format, try again. Enter *old_password/new_password/new_password* **to change your password.**

Explanation: The new password you entered is incorrect.

User Response: Try again.

IMW0240E Access denied. Unauthorized program loaded.

Explanation: You are running in an authorized environment, for example, BPX.DAEMON. This environment has been corrupted.

User Response: Determine how the environment has been corrupted and correct the problem.

IMW0241E Access denied - surrogate user setup error.

Explanation: An access control user ID is required.

User Response: Specify a valid access control user ID or surrogate user ID.

IMW0242E Access denied - system error using SAF.

Explanation: The user ID and password did not pass authentication.

User Response: Verify that the user ID and password are valid for this system.

IMW0243E Configuration file *name* **not found or in error. The server cannot start without a valid configuration file.**

Explanation: The server was either unable to find the specified configuration file, or the configuration file contained errors.

User Response: You might have entered the *name* incorrectly, or the configuration file may contain errors. Try again using a different name or check the file for errors.

IMW0244E Too many redirection-on-the-fly hops, max 10 (probably looping)

Explanation: More than 10 redirections have been issued; to avoid an infinite loop, the request processing has been ended.

User Response: Contact the server administrator.

IMW0245E Unable to PUT file: *filename filename*

Explanation: The attempted PUT operation has failed.

User Response: Contact the server administrator.

IMW0246E Unable to DELETE file: *filename filename*

Explanation: The attempted DELETE operation has failed.

User Response: Contact the server administrator.

IMW0247E Transfer-Encoding *transfer_encoding* **is not implemented.**

Explanation: An unknown transfer-encoding was applied to the body of the request.

User Response: Contact the client or proxy vendor.

IMW0248E Length Required.

Explanation: An HTTP/1.1 protocol violation has occurred. A PUT request was received and there was no transfer-encoding was applied. No content-length was specified. The server cannot determine the object size.

User Response: Contact the client or proxy vendor.

IMW0249E Input timer expired while serving client for request

Explanation: The request was not received from the client in the time specified on the InputTimeout directive.

User Response: Try again.

IMW0250E Output timer expired while serving client for request

Explanation: The request was not completed (sent to the client) in the time specified on the OutputTimeout directive.

User Response: Try again.

IMW0251E Script timer expired while serving client for request

Explanation: The CGI script failed to complete in the time specified on the ScriptTimeout directive.

User Response: Contact the script author or server administrator.

IMW0252E Timer of unspecified origin expired while serving client for request

Explanation: An internal timer expired.

User Response: None.

IMW0253E Redirection. This document can be found elsewhere. You see this message because your browser does not support automatic redirection handling.

Explanation: This page is issued when a redirection occurs for the benefit of those using browsers that do not understand redirection.

User Response: Follow the link.

IMW0254E Error

Explanation: This is the template for server generated error messages.

User Response: After you read the error, act accordingly.

IMW0255E Server mapping error. The server is misconfigured.

Explanation: The server cannot process the request because a value that it needs to proceed with the processing is not set up. This can be caused by a configuration error, such as a Pass directive matching a POST request. If you do not find a configuration error, this message may indicate a defect in the server.

User Response: Correct the configuration error.

If you cannot find the error, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0256E Can't open temporary PICS Configuration file: *filename* **(original was** *filename***)**

Explanation: The Web server address space was unable to open a temporary file previously created by the Web server daemon address space. Both the temporary file name and the original file names are provided in this message. Server processing has stopped. This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0257E Can't open temporary Config Configuration file: *filename* **(original was** *filename***), errno=***return_code***, errno2=***reason_code***.**

Explanation: The Web server address space was unable to open a temporary file previously created by the Web server daemon address space. Both the temporary file name and the original file names are provided in this message. Server processing has stopped. This is an internal error.

The z/OS UNIX System Services *return_code* and *reason_code* provide information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0258E Cannot open temporary protection setup file *filename*

Explanation: The Web server address space was unable to open a temporary file previously created by the Web server daemon address space. The temporary protection setup file name is provided in this message. Server processing has stopped. This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0259E Original file name *filename*

Explanation: The Web server address space was unable to open a temporary file previously created by the Web server daemon address space. This message gives the original protection file name (as it existed when the original configuration file was read). A previous message gave the temporary file name that could not be opened. The server address space has stopped. This is an internal error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0260E Cannot access configuration data, errno=*return_code*

Explanation: The Web server address space was unable to access configuration data that is created and maintained in shared memory by a daemon address space. The server address space has stopped. This is an internal error.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. Message IMW0261E or IMW0264E may also be issued with an z/OS UNIX System Services *reason_code* that can provide additional information. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0261I errno2=*reason_code*

Explanation: This message provides additional, z/OS UNIX-specific error information associated with a previous message.

The z/OS UNIX System Services *reason_code* provides information about the cause of the problem. For more information, see "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0262E WorkQueue... Unable to create configuration set, terminating.

Explanation: The daemon address space was unable to create a configuration set snapshot. Previous messages indicate the cause of the error.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0263E Cannot create shared memory, errno=*return_code*

Explanation: The daemon or server address space was unable to create a shared memory segment. The address space has stopped.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. Message IMW0261E or IMW0264E may also be issued with an z/OS UNIX System Services *reason_code* that can provide additional information. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0264E errno2=*reason_code*

Explanation: This message provides additional, z/OS UNIX-specific error information associated with a previous message.

The z/OS UNIX System Services *reason_code* provides information about the cause of the problem. For more information, see "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0265E Cannot open temporary file: *filename*

Explanation: When running with workload management, the daemon address space creates temporary files that reflect a snapshot of the configuration data as it existed when the daemon started. This file was successfully created, but could not be opened for writing. The daemon address space has stopped.

User Response: Determine the reason that the temporary file could not be opened for writing.

IMW0266E Cannot read configuration file: *filename*

Explanation: When running with workload management, the daemon address space creates temporary files that reflect a snapshot of the configuration data as it existed when the daemon started. Either the original configuration file could not be read, or the temporary file could not be written.

User Response: Determine the reason that the configuration file could not be read or the temporary file could not be written.

IMW0267E Request not valid -- HOST header was not sent

Explanation: This message indicates an HTTP/1.1 protocol violation. The Web server expected a host header that was not sent. This is a browser problem.

User Response: Contact the vendor of the browser.

IMW0268E Precondition failed: *request* has been modified.

Explanation: The object requested has been modified since the time specified in the If-Unmodified-Since header. You should not receive this message.

User Response: Contact the vendor of the browser.

IMW0269E Host not found or not responding

Explanation: An attempt was made to connect to the destination server that the client requested and failed.

User Response: Try again later.

IMW0270E Precondition failed: Could not match entity tags

Explanation: The version of the object does not match the version requested. You should not receive this message.

User Response: Contact the vendor of the browser.

IMW0271I Usage:

Explanation: This is a usage message for DPI run in test mode as a standalone process.

User Response: Try again after you correct the command syntax.

IMW0272I *command* **-d** *level* **-h** *hostname* **-c** *community* **-shm**

Explanation: This is a usage message for DPI run in test mode as a standalone process.

User Response: Try again after you correct the command syntax.

IMW0273I Where: -d level -debug level, default level=*number*

Explanation: This is a usage message for DPI run in test mode as a standalone process.

User Response: Try again after you correct the command syntax.

IMW0274I -h host -send request to specified host

Explanation: This is a usage message for DPI run in test mode as a standalone process.

User Response: Try again after you correct the command syntax.

IMW0275I -c community -use specified community name

Explanation: This is a usage message for DPI run in test mode as a standalone process.

User Response: Try again after you correct the command syntax.

IMW0276I -shm -connect over shared memory, not

Explanation: This is a usage message for DPI run in test mode as a standalone process.

User Response: Try again after you correct the command syntax.

IMW0277I Defaults: *command* **-d 0 -h** *host* **-c** *community*

Explanation: This is a usage message for DPI run in test mode as a standalone process.

User Response: Try again after you correct the command syntax.

IMW0278I *command* **-** *command*

Explanation: This is a usage message for DPI run in test mode as a standalone process.

User Response: Try again after you correct the command syntax.

IMW0279E DPI open failed with return code *return_code* **errno=***z/OS_UNIX_return_code***, errno2=***z/OS_UNIX_reason_code***.**

Explanation: The do_connect_open function failed. The Web server SNMP subagent cannot establish a connection with SNMP. SNMP support will not be successful.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Make sure:

- The SNMPD or DPID2 processes are running.
- The SNMP agent you are running is DPI capable and running with the DPI support on.
- The community name in the Web server configuration file permits SNMP to access this system.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0280E DPI do_register() failed with return code *return_code* **errno=***z/OS_UNIX_return_code***, errno2=***z/OS_UNIX_reason_code***.**

Explanation: The registration of the server SNMP MIB with the SNMP agent failed. SNMP support will not be successful.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Make sure the SNMP agent process is running.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0281E DPI dpi_process_request() failed with return code *return_code* **errno=***z/OS_UNIX_return_code*, **errno2=***z/OS_UNIX_reason_code***.**

Explanation: The server SNMP subagent could not process the DPI request. SNMP support will not be successful.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Make sure the SNMP agent is running.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0282E Problem getting a new DPI packet. rc=*return_code* **len=***number* **errno=***z/OS_UNIX_return_code*, **errno2=***z/OS_UNIX_reason_code*

Explanation: Retrieving a DPI packet using select, read, or DPIawait_packet_from_agent has failed with a bad return code, or has returned a DPI packet with a bad length. This DPI request processing has failed, but SNMP support will continue. The connection to the SNMP agent may be terminated.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Make sure the SNMP agent is running. The SNMP subagent attempts to reestablish the connection automatically.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0283E DPI cannot parse packet from agent

Explanation: The server SNMP subagent received a bad DPI packet. The parser DPIpacket returned a null header. SNMP support will continue if this failure happens after connect, open, and registration. If this fails during connect, open, or MIB registration to the SNMP agent, then SNMP subagent support will not continue.

User Response: Make sure the SNMP agent is running.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0284E Unexpected DPI packet type *type*

Explanation: Received a DPI packet that could be parsed, but the request type was not valid. The fulfillment of this request failed, but SNMP subagent support will continue.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0285E Processing the DPI request had an unexpected result, rc=*return_code***.**

Explanation: The DPI request type is valid, but fulfilling that request produced an error. This request fulfillment failed, but SNMP subagent processing will continue.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0286E DPI connect failed, errno=*return_code*, **errno2=***reason_code***.**

Explanation: The connection from the Web server SNMP subagent to the SNMP agent could not be established. SNMP support will not operate.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Make sure the SNMP agent is running, the SNMP agent is DPI capable, and the community name in the server configuration file is valid.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0287E DPI open failed

Explanation: The connection from the server SNMP subagent to the SNMP agent could not be established. SNMP support will not operate.

User Response: Make sure the SNMP agent is running, the SNMP agent is DPI capable, and the community name in the server configuration file is valid.

IMW0288E DPI send failed, rc=*return_code* errno=*z/OS_UNIX_return_code*, errno2=*z/OS_UNIX_reason_code*.

Explanation: The open packet could not be sent to the SNMP agent. SNMP support will not operate.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Make sure the SNMP agent is running, the SNMP agent is DPI capable, and the community name in the server configuration file is valid.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0289E DPI had a problem opening the subagent.

Explanation: A response to the open packet was not received from the SNMP agent. SNMP support will not operate.

User Response: Make sure the SNMP agent is running, the SNMP agent is DPI capable, and the community name in the server configuration file is valid.

IMW0290E DPI received responses with error.

Explanation: The SNMP agent sent the server SNMP subagent a DPI response with an error code in it during the connect and open or register process. SNMP support will not operate.

User Response: Make sure the SNMP agent is running, the SNMP agent is DPI capable, and the community name in the server configuration file is valid.

IMW0291E DPI cannot create an HTTP MIB registration packet.

Explanation: The connection from the Web server SNMP subagent to the SNMP agent could not be established. SNMP support will not operate.

User Response: Make sure the SNMP agent is running and the SNMP agent is DPI capable.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0292E DPI cannot send an HTTP MIB registration packet.

Explanation: SNMP subagent could not send the registration packet for the server MIB to SNMP. SNMP support will not operate.

User Response: Make sure the SNMP agent is running and the SNMP agent is DPI capable.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0293E DPI received no reply to an HTTP MIB registration packet.

Explanation: SNMP subagent did not receive a response to the registration packet for the Web server MIB from the SNMP agent. SNMP support will not operate.

User Response: Make sure the SNMP agent is running and the SNMP agent is DPI capable.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0294E DPI response not of RESPONSE type.

Explanation: The SNMP subagent expected a response type packet during connect and open or register and did not receive a packet of RESPONSE type. SNMP support will not operate.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0295E DPI cannot create the Application MIB register packet.

Explanation: The SNMP subagent could not create a registration packet for the Web server MIB. SNMP support will not operate.

User Response: Make sure the SNMP agent is running and the SNMP agent is DPI capable.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0296E DPI cannot send the Application MIB register packet.

Explanation: The SNMP subagent could not send the registration packet for the Web server MIB to SNMP. SNMP support will not operate.

User Response: Make sure the SNMP agent is running and the SNMP agent is DPI capable.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0297E DPI received no reply to the Application MIB register packet.

Explanation: The SNMP subagent did not receive a response to the registration packet for the Web server MIB from the SNMP agent. SNMP support will not operate.

User Response: Make sure the SNMP agent is running and the SNMP agent is DPI capable.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0298E DPI Error: For instance ???

Explanation: During GET or GET NEXT processing, the variable list from the mkDPIset API was returned as null. This GET request failed, but SNMP subagent support will continue.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0299E DPI Error: varBind_p is null

Explanation: During GET or GET NEXT processing, a variable bind structure in a list was null. This GET request failed, but SNMP subagent support will continue.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0300E DPI cannot create GET Response packet

Explanation: During GET processing, mkDPIresponse returned a null response packet. This GET request failed, but SNMP subagent support will continue.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0301E SNMP open/registration failed, going into retries

Explanation: The server's SNMP subagent is trying to contact the SNMP master agent on the host. It has timed out or an error has been received. The SNMP subagent will begin trying session establishment on an increasing interval (1 minute, 2 minutes, 4 minutes, 8 minutes...). SNMP MIB values for the server will not be available.

User Response: Make sure the SNMP master agent is running and responding to SNMP requests.

IMW0302E SNMP: No local host name could be found, domain name will be used.

Explanation: The gethostbyname function could not look up the hostname, so the hostname returned by gethostname is used, which does not have a domain name attached to it.

User Response: Set the local host name on your system.

IMW0303I An SNMP thread is already running.

Explanation: SNMP subagent thread was running, so another one could not be started. This happens when the server is issued a restart and the SNMP request is already operational.

User Response: None.

IMW0304E The server is not permitted to the BPX.SMF facility class and cannot record SMP records.

Explanation: RACF permission has not been set up to allow the server to record SMF records.

User Response: Authorize the user ID that your Web server is running under to have read authority to the BPX.SMF facility class in RACF.

IMW0305E SMF record length was too big, Server performance record not written.

Explanation: The server performance record is too long and could not be written.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0306E Received unexpected error from SMF trying to write Server Performance record, record not written.

Explanation: An error was received from SMF. The error is not documented.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0307E SMF is not accepting Server Performance records at this time (type 103, subtype 02).

Explanation: SMF has not been enabled to record record type 103, subtype 02 and has rejected the record.

User Response: Your system programmer should:

- Change the SMF parameters so that SMF is enabled to write SMF records type 103, subtype 02
- Start SMF

IMW0308E SMF is not active, Server Performance record not written

Explanation: The SMF application is not running. The performance record does not get written.

User Response: Have your system programmer or operations center start SMF.

IMW0309E Received error from SMF trying to write Server Performance record, record not written.

Explanation: An error was received by the server from SMF. SMF is running and is accepting performance records. The record is accurate.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0310E SMF record length was too big, Server Configuration record not written.

Explanation: The server configuration record is too long and could not be written.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0311E SMF is not accepting Server Configuration records at this time (type 103, subtype 01).

Explanation: SMF has not been enabled to record record type 103, subtype 01 and has rejected the record.

User Response: Have your system programmer change the SMF parameters so that SMF is enabled to write SMF records type 103, subtype 01.

IMW0312E SMF is not active, Server Configuration record not written.

Explanation: The SMF application is not running. The configuration record was not be written.

User Response: Have your system programmer or operations center start SMF.

IMW0313E Received error from SMF trying to write Server Configuration record, record not written.

Explanation: An error was received by the server from SMF. SMF is running and is accepting performance records, and the record is accurate.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0314E Received unexpected error from SMF trying to write Server Configuration record, record not written.

Explanation: An error was received from SMF that is not documented.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0315E WorkQueue... initialization failed, terminating.

Explanation: A _server_init() request to workload management was made and failed.

User Response: Determine the reason for this error:

- Use _errno2() to obtain the workload management service reason for the failure.
- The calling thread's address space is not permitted to the BPX.WLMSERVER facility class.

IMW0316I WorkQueue... connecting to WLM as *queue_manager* **or** *queue_server*, **SN:** *subsystem_name*, **AE:** *application_environment*, **EU:** *execution_units*

Explanation: This message indicates which address space is being connected to workload management and relevant information about queue manager and queue server.

User Response: None.

IMW0317E Invalid request -- completely unable to parse it

Explanation: A request line was sent to the parser and it is not understood. A 400 message is returned to the server.

IMW0318E Error: Parameter *parameter* specified more than once

Explanation: A parameter has been specified more than once on the passed parameters. -SN and -AE must be specified only once per workload management address space.

User Response: Correct your parameters and resubmit the job.

IMW0319E Missing value for *parameters* parameters

Explanation: The value for the specified parameter is missing (-SN or _AE).

User Response: Be sure you are using the appropriate syntax, update your job, and resubmit it.

IMW0320E Parameter conflict. Only -SN is allowed with -AE.

Explanation: Additional parameters were found when trying to verify the queue server address space. The only ones accepted are -SN and -AE.

User Response: Be sure you are using the appropriate syntax, update your job, and resubmit it.

IMW0321E Error: Parameter -SN required with -AE

Explanation: A subsystem name is required with the ApplEnv parameter.

User Response: Be sure you are using the appropriate syntax, update your job, and resubmit it.

IMW0322E Connection established

Explanation: The client has been notified that a connection was established.

User Response: None.

IMW0323E Application Environment currently not available

Explanation: The ApplEnv directive for a URL does not specify a correctly configured WLM Application Environment.

User Response: Ensure that the ApplEnv directive for this URL specifies a correctly configured WLM Application Environment, and check the status of the WLM Application Environment associated with this request.

For additional information, check the error log for message IMW0563E. IMW0563E contains the z/OS UNIX System Services errno (return code) and errno2 (reason code) of the failing WLM operation. IMW0563E can also been seen in the Web server –vv trace. To find the message in the trace, start the Web server with the -vv trace option and recreate the failure. Examine the trace output to find the IMW0563E message along with the error message, the return code and the reason code associated with the failure. The trace message includes text similar to "IMW0563E WorkQueue... *WLM function_name* failed; errno=*return_code*; errno2=*reason_code*; errmsg: *explanation of errno*". See "Explanation of errno and errno2 codes in messages" on page 605.

IMW0324E Expectation failed

Explanation: A client browser sent an expect header that could not be satisfied by the HTTP server. The server returns a 417 Failed Response message if the expect header has data in it that wasn't recognized by the server if the server doesn't support expect headers.

User Response: None

IMW0325E Accept headers not matched

Explanation: This message, which is written to the log file, is equivalent to IMW0326E.

User Response: None

IMW0326E Not Acceptable - No file exists which can satisfy the accept headers sent with the request.

Explanation: Client accept headers can identify various file characteristics (for example: mime-type, language, encoding type, compression, character set, or user agent). The browser requested a file with specific characteristics and no file with those characteristics was found on the server.

User Response: Ensure the client and server are properly configured. For example, requesting a file based on language or character sets, make sure the client and browser are configured appropriately.

If accompanied by a 406 response, the file was found on the server, but it did not match one or more of the criteria specified in the accept header.

IMW0327E The script request is not valid. The CGI file cannot be found or is not executable.

Explanation: You have requested that a CGI be executed, but the CGI file is either not found or it cannot be executed for some reason.

User Response: This may be a permission bit problem. Contact your System Administrator to correct the error with the CGI file.

IMW0328E Can't browse selected file.

Explanation: The requested file has been found, but could not be opened. You may not have permission to view the file.

User Response: If you believe you should be given permission to access the file, contact your System Administrator to correct the problem.

IMW0329E Completion code from REXX exec *exec_name* is *completion_code*.

Explanation: When a GWAPI REXX executable program exits, it must set a numeric completion code with the EXIT instruction. This message is issued when the exec completion code is outside the acceptable HTTP return code range of -1 to 599.

User Response: Correct the error.

IMW0330E Error *error_code* attempting to execute *exec_name*.

Explanation: A problem occurred attempting to invoke the GWAPI REXX executable program *exec_name*.

error_code is one of the following values:

–99 A storage request failed.

–98 Necessary information, such as a path name or exec name, could not be extracted from the server variables.

–97 Problems occurred processing the executable program file. Possible problems include non-existent file, improper permission, or no free file descriptor.

–96 The executable program contained more than 16,384 lines.

–95 BPXWRBLD service which creates z/OS REXX environment failed.

User Response: Correct the error.

IMW0332E Can't obtain shared memory for connection pool, errno=*return_code*.

Explanation: The server cannot obtain the requested shared memory.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Modify the connection pool parameters, or ask the system programmer to increase the amount of shared memory that can be allocated on the system.

IMW0333E Missing values on -ss parameter.

Explanation: The -ss parameter was specified without the necessary number of additional arguments.

User Response: Restart the server specifying the -ss parameter with the correct number of arguments. The format of the -ss parameter is: -ss CONNECTION_POOL start_address size.

IMW0334E More than one set of values specified for CONNECTION_POOL. Last set used

Explanation: The -ss parameter was specified with multiple CONNECTION_POOLS. This is not currently supported. The last set CONNECTION_POOL argument was used.

User Response: The server was started using the last CONNECTION_POOL argument specified. If incorrect, stop the server, then restart the server using the correct parameters.

IMW0335I Missing "}". Incomplete ApplEnvConfig statement.

Explanation: The end of the configuration file was reached without finding the ending } of the ApplEnvConfig directive.

User Response: Correct the configuration file.

The correct syntax is:

```
ApplEnvConfig AEName {
  directive
  directive
:
  }
```

IMW0336I Multiple conflicting PluginDefault statements

Explanation: There were conflicting PluginDefault statements in the configuration file. PluginDefault Include and PluginDefault Exclude cannot both be specified at the same level.

User Response: Correct the configuration file.

IMW0337I Multiple conflicting Plugin directives for this DLL: string

Explanation: There were conflicting PluginInclude and PluginExclude statements in the configuration file for the same DLL at the same level. PluginInclude and PluginExclude cannot both be specified at the same level for the same DLL.

User Response: Correct the configuration file.

IMW0338I Invalid parameter to PluginDefault directive: string

Explanation: An invalid value was specified on the PlugingDefault Directive.

User Response: Correct the configuration file. Valid values are Include and Exclude.

IMW0339I Syntax error in configuration file, expecting '{' to start ApplEnvConfig statement for string

Explanation: The ApplEnvConfig directive in the configuration file is not valid.

User Response: Correct the configuration file. The correct syntax is:

```
ApplEnvConfig AEName {
  directive
  directive
⋮
  }
```

IMW0340I Active threads reset from *number* to system limit of *number*

Explanation: The number of threads specified in the configuration file exceeded the maximum number of threads per process allowed by the system.

User Response: To eliminate this warning message, reduce the maximum number of threads in the configuration file to less than the limit imposed by the operating system.

IMW0341E Value for EnableFRCA is not valid: *value*.

Explanation: The value specified for the EnableFRCA directive is not valid. This directive is used to turn the Fast Response Cache Accelerator dynamic caching function on or off.

For more information, see "EnableFRCA — Turn dynamic caching on or off" on page 567.

User Response: Correct the configuration file.

IMW0342E FRCAStackName is not valid: *name*.

Explanation: The TCP/IP stack name specified on the FRCAStackName directive is not valid.

For more information, see "FRCAStackName — Specify the TCP/IP stack that supports the dynamic cache" on page 571.

User Response: Correct the configuration file.

IMW0343E FRCAWLMParms parameter is not valid: *parameter*.

Explanation: The parameter specified on the FRCAWLMParms directive is not valid. This directive is used to specify the unique subsystem name, application environment name, and transaction class that will be used to classify the work performed by the Fast Response Cache Accelerator under Workload Management (WLM).

For more information, see "FRCAWLMParms — Specify parameters for Workload Management" on page 572.

User Response: Correct the configuration file.

IMW0344E Both FRCACacheOnly and FRCANoCaching cannot be used.

Explanation: You must use either the FRCACacheOnly and FRCANoCaching directive.

For more information, see "FRCACacheOnly — Specify URIs to be dynamically cached" on page 570 and "FRCANoCaching — Exclude URIs from the dynamic cache" on page 571.

User Response: Correct the configuration file.

IMW0345E Fast Response Cache Accelerator did not initialize.

Explanation: This message can occur as a result of system errors or user setup errors.

User Response: Check the error log for the z/OS UNIX System Services errno (return code) and errno2 (reason code) related to _server_pwu() or WorkQueue operations. If you do not see these codes in the error log, start the Web server with the -vv trace option and examine the trace output to find the return and reason codes associated with the failure. See "Explanation of errno and errno2 codes in messages" on page 605.

IMW0346E Debug Tool start had failure code: *failure_code*.

Explanation: While attempting to start the z/OS Debug Tool, a failure occurred that prevented the tool from starting. Message IMW0347E is also issued and shows the GWAPI plugin module name.

The failure code is usually one of the following values:

CEE000
: Non-zero return code from CEE3CBTS or Dflow_allocation failure

CEE2F2
: The Debug Tool is not available.

CEE2F7
: Profiler loaded, but Debug Tool is not available.

User Response: Verify that the Remote Debugger

workstation has been started. If the Remote Debugger workstation is up and running, verify that the DebugToolAddr directive in your httpd.conf file specifies the correct address and port number for the Remote Debugger workstation.

Related information:
- "DebugToolAddr - Identify the workstation running the Remote Debugger" on page 485
- "Debugging C/C++ GWAPI programs" on page 375

IMW0347E start_dbg() could not start LE Debug Tool for *GWAPI_plugin_module_name*.

Explanation: While attempting to start the LE Debug Tool, a failure occurred that prevented the tool from starting. Message IMW0346E is also issued and provides information on the cause of the problem.

User Response: See the description of message IMW0346E.

IMW0348E DebugToolAddr directive had gethostbyname error.

Explanation: During configuration processing, the host name for the Remote Debugger workstation could not be resolved.

User Response: Verify that the host name specified on the DebugToolAddr directive is correct and that the host name is recognized by your Domain Name Server. You can specify the IP address of the Remote Debugger instead of the host name on this directive.

For more information, see "DebugToolAddr - Identify the workstation running the Remote Debugger" on page 485.

IMW0349E DebugToolAddr directive had PORT parameter error.

Explanation: The port number value specified for the LE Debug Tool Remote Debugger workstation is not valid. The port number must be an integer containing numbers 0-9.

User Response: Correct the port number on the DebugToolAddr directive.

For more information, see "DebugToolAddr - Identify the workstation running the Remote Debugger" on page 485.

IMW0350I Storage allocation error for InternalTraceLog. Using default size.

Explanation: The amount of storage requested for the Web server Internal Trace on the InternalTraceLog directive cannot be allocated. The Web server will attempt to use the default Internal Trace log size.

User Response: For guidance and additional information on Web server traces, contact the IBM Software Support Center. For support information on the Web, see "Support services and resources" on page 604.

IMW0351E Unable to allocate *number* KB. No InternalTraceLog available.

Explanation: This error occurs when storage cannot be allocated for the Web server Internal Trace log. This message usually indicates that storage for the default Internal Trace size could not be allocated. Message IMW0350I is also issued.

User Response: You may need to increase the region size. See Message IMW0350I for additional information.

IMW0352E InternalTraceSize error - invalid characters in size.

Explanation: This error occurs when the InternalTraceSize directive contains characters that are not valid. The trace size value must be an integer containing numbers 0-9.

User Response: For guidance and additional information on Web server traces, contact the IBM Software Support Center. For support information on the Web, see "Support services and resources" on page 604.

IMW0353E Failure waiting for I/O completion, errno = *return_code*.

Explanation: A signal timed wait for a previously issued asynchronous I/O request failed. As a result of this condition, I/O from the client may be lost, and the connection will eventually be closed when the proper timeout (input, output, or persistent) is reached.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. Message IMW0261E or IMW0264E may also be issued with an z/OS UNIX System Services *reason_code* that can provide additional information. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0354E Immediate failure on asynch read operation for socket *socket_number*, errno = *return_code*.

Explanation: This failure indicates a system resource constraint or an error in the Web server. It occurs when an asynchronous I/O request to read data on *socket_number* fails. This failure occurs immediately on the issuance of the asynchronous I/O.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. Message

IMW0261E or IMW0264E may also be issued with an z/OS UNIX System Services *reason_code* that can provide additional information. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0355E Immediate failure on asynch write operation for socket *socket_number*, **errno =** *return_code*.

Explanation: This failure indicates a system resource constraint or an error in the Web server. It occurs when an asynchronous I/O request to write data on *socket_number* fails. This failure occurs immediately, that is, on the issuance of the asynchronous I/O.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. Message IMW0261E or IMW0264E may also be issued with an z/OS UNIX System Services *reason_code* that can provide additional information. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0356E Data in authorization header not valid: *string*.

Explanation: Browser credentials need to be updated.

User Response: Start a new browser and enter a user ID and password with proper credentials.

IMW0357E Data in content-range header not valid for content-range *string*.

Explanation: A valid content-range header from a client contains information on the data type, location, and length, for example:

```
Content_Range: bytes 734-1233/1234
```

This message is issued when the Web server detects an error in the content-range header information sent by a client. Possible reasons for the error are:

- The TYPE field is either missing or is not valid.
- The LOCATION/LENGTH range information is not complete.
- A separator is not valid; only hyphens (-) and slashes (/) are valid.
- Separators are valid but in the wrong order.
- An ending location is smaller than the beginning location.
- An ending location is greater than the length.

The incorrect content-range information was probably generated by a client applet. The error can also be caused by an I/O error on a socket.

User Response: Correct the applet which generated the incorrect content-range information.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0358E Unknown expect header: *string*

Explanation: The only supported expect header is `100-CONTINUE`.

User Response: Correct the expect header or contact the browser vendor.

IMW0359E Unknown pragma header: *string*

Explanation: The only supported pragma header is `NO-CACHE`.

User Response: Correct the pragma header or contact the browser vendor.

IMW0360E Data in proxy authorization header not valid: *string*.

Explanation: Browser credentials need to be updated.

User Response: Start a new browser and enter a user ID and password with proper credentials.

IMW0361E Unknown or unexpected header: *string*.

Explanation: The header is not recognized by the Web server.

User Response: Correct the header or contact the browser vendor.

IMW0362E FastCGI error: failed to open *configuration_file*, **errno=***return_code*, **errno2=***reason_code*.

Explanation: The FastCGI configuration file could not be opened during FastCGI initialization.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0363E FastCGI initialization failed: no FastCGI applications defined.

Explanation: No Local or External FastCGI applications are defined in the FastCGI configuration file.

User Response: Correct the FastCGI configuration file.

IMW0364E FastCGI initialization failed, errno=*return_code*, errno2=*reason_code*.

Explanation: FastCGI processes could not be created during FastCGI initialization.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0365E FastCGI initialization failed to establish signal handler for *signal_number*, errno=*return_code*, errno2=*reason_code*.

Explanation: A signal handler could not be created during FastCGI initialization.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0368E FastCGI error: could not create socket, errno=*return_code*, errno2=*reason_code*.

Explanation: The request cannot be handled because FastCGI could not create a TCP/IP socket to connect to a FastCGI process.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0369E FastCGI error: could not set socket option *option*, errno=*return_code*, errno2=*reason_code*.

Explanation: FastCGI could not set the indicated TCP/IP socket option.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0370E FastCGI error: failed to connect to FastCGI process, errno=*return_code*, errno2=*reason_code*.

Explanation: This error usually occurs when you attempt to run a FastCGI program and have logged into the browser with the same user ID used to start the Web server. In addition to this message, you may get a browser Error 503.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Log into the browser with a user ID that is different from the user ID you used to start the Web server.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0371E FastCGI error: write request failed, errno=*return_code*, errno2=*reason_code*.

Explanation: FastCGI cannot write the request to the FastCGI process.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0372E FastCGI error: could not retrieve Web server environment variables.

Explanation: FastCGI cannot retrieve environment variables from the Web server.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information

on the Web, see "Support services and resources" on page 604.

IMW0373E FastCGI error: attempt to read request failed, content length: *string***.**

Explanation: FastCGI cannot read the request body. Possible causes are:
- The `Stop` option was selected on the browser before a request completed.
- The request body is not valid.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0374E FastCGI initialization error: failed to create *socket_number* **socket returning 0, errno=***return_code***, errno2=***reason_code***.**

Explanation: During FastCGI initialization, communication between the FastCGI process and the Web server could not be established.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0375E FastCGI initialization error: failed to bind to *socket_number* **socket, errno=***return_code***, errno2=***reason_code***.**

Explanation: Bind to TCP/IP socket failed during FastCGI initialization.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0376E FastCGI initialization error: unlink failed on *directory_entry***, errno=***return_code***, errno2=***reason_code***.**

Explanation: Unlinking the AF_UNIX directory entry failed during FastCGI initialization.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0377E FastCGI initialization error: failed to set correct permission bits for *directory_entry***, errno=***return_code***, errno2=***reason_code***.**

Explanation: During FastCGI initialization, correct permission bits were not set for the AF_UNIX directory entry.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0378E FastCGI initialization error: failed to get host name *host_name***, errno=** *return_code***, errno2=** *reason_code***.**

Explanation: During FastCGI initialization, *host_name* could not be found.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0379E FastCGI error: failed to read data, errno=*return_code***, errno2=***reason_code***.**

Explanation: FastCGI attempted to read the response from the FastCGI application, but no data was received.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0380E FastCGI error: failed to find matching entry for the request, errno=*return_code***, errno2=***reason_code***.**

Explanation: FastCGI could not find a matching entry for the request in the FastCGI configuration file.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Ensure that the URL directive in the FastCGI configuration file matches the FastCGI request.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0381E CGI error: no script name in the request.

Explanation: The CGI script cannot be run because there is no script name in the request.

User Response: Ensure that the request contains a script name and resubmit.

IMW0382E CGI read error: failed to read CGI script, return code =*code***, errno=***return_code***, errno2=***reason_code***.**

Explanation: A failure occurred while reading the CGI script results. No data was read.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0383E Internal read error: failed to read CGI script, errno=*return_code***, errno2=***reason_code***.**

Explanation: The fdopen() failed to open a pipe for reading CGI results. No data was read.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0384E CGI read error: failed to read CGI script, errno=*return_code***, errno2=***reason_code***.**

Explanation: CGI script results cannot be read.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0385E SNMP error: buffer not large enough for new DPI packet. Buffer = *number_of_bytes***, packet =** *number_of_bytes***, return code =***code***, length =** *number***.**

Explanation: The size of a DPI packet exceeded the size of the buffer. DPI request processing has failed, but SNMP support will continue.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Ensure that the SNMP agent is running. The SNMP subagent automatically attempts to reestablish the connection.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0386E Initialization error: Protection setup file *filename* **not read.**

Explanation: A Protect directive referenced a protection setup file that could not be opened by the Web server.

User Response: Verify:
- The protection setup file exists.
- The filename is spelled correctly in the Web server configuration file.
- The file has the correct permissions.

Correct the configuration file or protection setup file.

IMW0387E Caching error: failed to open file *filename***, URL =***URL***, thread =** *number***, errno =** *return_code***.**

Explanation: The RequestToCache operation failed.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0388E Internal I/O error: unable to load file contents to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0389E Internal HTLoadCacheToStream error: unable to load file contents from cache to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0390E Internal HTLoadToStream error: unable to load proxy result to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0391E Internal HTLoadToStream error: unable to load proxy result to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0392E Internal HTLoadToStream error: unable to load proxy file to the source server, errno=*return_code*, errno2=*reason_code*.

Explanation: The proxy request cannot be written to the source server.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0393E CGI write error: failed to write CGI output to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0394E Internal HTLoadStreamFromRamCache error: unable to load file contents from cache to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0395E Internal I/O error: unable to copy file contents to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information

on the Web, see "Support services and resources" on page 604.

IMW0396E Internal HTLoadScriptResultSSI error: unable to write file contents to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred while attempting to write the result of an SSI script to the browser.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0397E Caching error: failed to get statistics for *filename*, URL =*URL*, thread = *number*, errno = *return_code*.

Explanation: The RequestToCache operation failed.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0398E Caching storage request failed: unable to allocate *number* bytes, thread = *number*, URL = *URL*, errno = *return_code*.

Explanation: A request for storage failed.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0399E Caching error: failed to read file *filename*, thread = *number*, errno = *return_code*.

Explanation: File cannot be read.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0400E GWAPI HTLoadScriptResultAPI error: unable to write file contents to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred while attempting to write the result of a script called from a GWAPI to the browser.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0401E GWAPI HTLoadStreamFromRamCache error: unable to write file contents from cache to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred in the GWAPI.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0402E GWAPI error: unable to copy file contents to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred in the GWAPI.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0403E GWAPI write error: unable to write results to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred in the GWAPI.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about

the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0404E GWAPI writeCP error: unable to write results to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0405E Internal I/O error: unable to write file contents to the browser, errno=*return_code*, errno2=*reason_code*.

Explanation: An I/O error occurred.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0406E Failed to load file *filename* in Fast Response Cache Accelerator cache, thread = *number*, errno =*return_code*.

Explanation: File cannot be loaded in the cache.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0407E Failed to unload file *filename* from Fast Response Cache Accelerator cache, thread = *number*, errno = *return_code*.

Explanation: File cannot be unloaded from the cache.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0408E Failed to register *filename* in Fast Response Cache Accelerator cache, errno=*return_code*.

Explanation: Internal error. File cannot be registered in the cache.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0409E Fast Response Cache Accelerator did not initialize, errno=*return_code*, errno2=*reason_code*.

Explanation: Initialization failed.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0410E Internal error: global lock request failed, errno=*return_code*.

Explanation: An internal error occurred when the Web server attempted to remove a shared memory semaphore. A global lock is required for exclusive write access to the memory segment.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0411E Internal error: global unlock request failed, errno=*return_code*.

Explanation: An internal error occurred when the Web server attempted to remove a shared memory semaphore.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0412E Caching error: unable to locate file in the cache, errno=*return_code*, errno2=*reason_code*.

Explanation: File cannot be opened in the cache.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0413E PROXY AUTHENTICATE REQUIRED

Explanation: Proxy authentication failed because the browser does not support it. Additional messages may be issued.

User Response: None.

IMW0414E PROXY UNAUTHORIZED

Explanation: Not authorized for proxy access to the document. Additional messages may be issued.

User Response: None.

IMW0415E PASSWORD EXPIRED

Explanation: Access has been denied because your password has expired.

User Response: If you do not have authorization to change an expired password, contact your System Administrator.

IMW0416E PROXY PASSWORD EXPIRED

Explanation: Access has been denied because your password has expired.

User Response: If you do not have authorization to change an expired password, contact your System Administrator.

IMW0417E PASSWORD CHANGED

Explanation: This message confirms that the password has been changed.

User Response: None.

IMW0418E PROXY PASSWORD CHANGED

Explanation: This message confirms that the password has been changed.

User Response: None.

IMW0419E UNEQUAL NEW PASSWORD

Explanation: The new passwords do not match.

User Response: Check and reenter the passwords.

IMW0420E PROXY UNEQUAL NEW PASSWORD

Explanation: The new passwords do not match.

User Response: Check and reenter the passwords.

IMW0421E NEW PASSWORD FORMAT IS NOT VALID

Explanation: The password entered is in a format that is not valid, or an incorrect password has been entered more than the allowed number of times.

User Response: Try to enter the password again. If you continue to have problems, contact your System Administrator.

IMW0422E PROXY NEW PASSWORD FORMAT IS NOT VALID

Explanation: The password entered is in a format that is not valid, or an incorrect password has been entered more than the allowed number of times.

User Response: Try to enter the password again. If you continue to have problems, contact your System Administrator.

IMW0423E ADDRESS SPACE DIRTY

Explanation: An unauthorized program load occurred.

User Response: Review the program control requirements for the Web server in "Completing and customizing your Web server installation" on page 20.

IMW0424E SETUP ERROR — SURROGATE SUPPORT

Explanation: An error was detected in a surrogate user ID setup.

User Response: Review the guidelines for setting up surrogate user IDs in "Completing and customizing your Web server installation" on page 20.

IMW0425E INTERNAL ERROR — ACCESS CONTROL FAIL

Explanation: An internal error or unknown SAF error has occurred.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0426E FAILURE CODE BY API

Explanation: A return code other than 0 (no action) or 200 (ok) was returned by the Authorization exit.

User Response: None.

IMW0427E MULTI REDIRECTION

Explanation: The request has been redirected.

User Response: None.

IMW0429E Cannot start Java runtime for trusted servlets, rc=*return_code*

Explanation: An error occurred when the Web server attempted to start the Java™ runtime environment. Possible causes are:

- The Web server could not determine the local hostname.
- The Web server was not able to load the Java DLL or find a Java function. A message issued prior to this message may provide more information.
- The Web server was not able to construct the CLASSPATH for the Java trusted servlet directory.

User Response: Correct the error and retry.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0430E Cannot start Java runtime for servlets, rc=*return_code*

Explanation: An error occurred when the Web server attempted to start the Java™ runtime environment. Possible causes are:

- The Web server could not determine the local hostname.
- The Web server was not able to open a server socket or to accept a session.
- The Web server was not able to construct the CLASSPATH for the Java servlet directory.
- The Web server was not able to run fork() or exec() because of a system problem.

User Response: Correct the error and retry.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0431E Could not restart Java runtime for trusted servlets.

Explanation: An error occurred during a restart of the Web server. The Web server sent a restart request to the Java™ Virtual Machine (JVM), and the request failed.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0432E Could not restart Java runtime for servlets.

Explanation: An error occurred during a restart of the Web server. The Web server sent a terminate request to the Java™ Virtual Machine (JVM), and the request failed.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0433E Can't find *zip_file* in the CLASSPATH.

Explanation: The Web server could not find the zip file containing the class to be executed.

User Response: Modify the CLASSPATH environment variable to permit access to the required zip file and retry.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0434E ServletTrustedDir configuration directive must be set. Java runtime will not be started for trusted servlets.

Explanation: An error occurred when the Web server attempted to start the Java™ runtime environment.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0435E ServletDir configuration directive must be set. Java runtime will not be started for servlets.

Explanation: An error occurred when the Web server attempted to start the Java™ runtime environment.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0436E Return code *return_code* loading DLL module *name*.

Explanation: An error occurred when the Web server attempted to load the DLL in this message. Possible causes are:

- The DLL does not exist.
- The DLL is not available in the path specified on the LIBPATH environment variable.
- The DLL is not needed because of a PluginExclude directive setting. If this is the cause, you will see a message similar to the following message in the Web server trace: `ApplEnvFilter...Not loading dll due to ApplEnv PluginExclude`.
- There is a permission bit problem.

User Response: If the DLL is needed, ensure that the LIBPATH environment variable in the Web server httpd.envvars file contains the correct path for the DLL. Also ensure that the permission bit setting allows the Web server to execute the DLL.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0437E Return code *return_code* loading function *function_name* from DLL module *name*.

Explanation: An error occurred when the Web server attempted to load the function in this message. This function was not found in the DLL.

User Response: Verify that the DLL contains the function.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0438E Serverinit Error: server did not load functions from DLL module *name*

Explanation: An error occurred when the Web server attempted to load a function from the DLL in this message.

User Response: Messages issued prior to this message will provide the name of the function and recommended actions.

IMW0439E Cannot create the cache information table, errno=*return code*, errno2=*reason_code*

Explanation: An error occurred when the Web server attempted to lock a cache information table. Caching has been turned off.

The z/OS UNIX System Services *return_code* and *reason_code* provide information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0440E Cannot open the temp servlet rule file: *file_name*

Explanation: An error occurred when the Web server attempted to open the servlet rule file for output.

User Response: Check the permission bits corresponding to the path and name of the file.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0441E Could not include LDAP configuration: *LDAPInclude_ statement*

Explanation: The Web server LDAPInclude statement listed in this message was used to retrieve configuration information from an LDAP server, but the retrieval was not successful.

This message can be issued if there is no information stored on the LDAP server, or if the LDAPInfo-label, search-filter, or attribute fields is not coded correctly.

User Response: Check the information in the LDAP directory to make sure the information is available on the server. If the information is available, ensure that the LDAPInfo-label, search-filter, and attribute fields are coded correctly.

IMW0442E Syntax error in LDAPInfo configuration for ldap server *server_name*

Explanation: There is an error in the Web server LDAPInfo definition. The error is described in a previous message.

User Response: This message may be preceded by one or more Web server messages in the range IMW0504W-IMW0531W. These messages provide more information on configuration file syntax errors for LDAP.

If you cannot find the error, contact the IBM Software Support Center for assistance. For support information

on the Web, see "Support services and resources" on page 604.

IMW0443E Maximum number of cipher specifications in SSLCipherSpec directive exceeded.

Explanation: More than 64 cipher specifications were specified on SSLCipherSpec directives in the Web server configuration file, httpd.conf..

User Response: Reduce the number of cipher specifications.

For more information and examples, see "SSLCipherSpec - Specify the levels of encryption to use for SSL connections" on page 547.

IMW0444E SSL encryption method not supported.

Explanation: A cipher specification that is not supported by the Web server is specified on a Web server SSLCipherSpec directive.

User Response: Correct the cipher specification that is not valid.

For information on supported cipher specifications, see "SSLCipherSpec - Specify the levels of encryption to use for SSL connections" on page 547.

IMW0445E The method you tried to disable is not valid: *method*

Explanation: The method specified on the -disable option on the Web server startup command line is not valid.

User Response: Retry the command using a valid method.

IMW0446E The parameter to -normal_mode option is not valid: *parameter*

Explanation: The parameter specified on the -normalmode option on the Web server startup command is not valid. This error can also occur if the value specified on the Web server's NormalMode directive is not valid. Valid values are `on` or `off`.

User Response: Retry the command using a valid parameter or correct the value on the NormalMode directive.

IMW0447E The parameter to -p option is not valid: *parameter*

Explanation: The parameter specified on the -p or -sslport option of the Web server startup command must be an integer.

User Response: Retry the command and specify an integer.

IMW0448E SSL label lookup failed.

Explanation: An attempt to retrieve a key from a key database failed.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

If IBM Software Support requests a trace, the trace will identify the name of the key.

IMW0449E Cannot change process group.

Explanation: The Web server could not change its process group. Processing continues.

User Response: None.

IMW0450E Cannot fork second child.

Explanation: The Web server attempted to create a new process, but executed the fork() command unsuccessfully. Processing continues.

User Response: None.

IMW0451E Cannot get group entry for *name*

Explanation: The name specified on a GroupFile subdirective in a protection setup was not valid.

User Response: Correct the GroupFile subdirective on the Protection directive.

IMW0452E Cannot fork to go background.

Explanation: An internal Web server error occurred.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0453E Security not available. *module_name or function name* **did not load.**

Explanation: An internal Web server error occurred.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0454E SIGHUP caught - waiting for threads to drain.

Explanation: An internal Web server error occurred.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0455E Immediate failure on asynch write operation for socket *socket_number*, **errno =** *return_code*.

Explanation: This failure indicates a system resource constraint or an error in the Web server. It occurs when an asynchronous I/O request to write data on *socket_number* fails. This failure occurs immediately, that is, on the issuance of the asynchronous I/O.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. Message IMW0261E or IMW0264E may also be issued with an z/OS UNIX System Services *reason_code* that can provide additional information. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0456E Error obtaining key for configuration, errno=*return_code*

Explanation: An internal Web server error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0457E Unable to open file *file_name* **errno=***return_code*

Explanation: An error occurred when the Web server attempted to open *file_name*.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0458E Unable to read or write file *file_name*, **errno=***return_code*.

Explanation: An error occurred when the Web server attempted to read or write *file_name*.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0459E Unable to obtain key, errno=*return_code*

Explanation: This error can occur if the WLM subsystem name specified on the -SN parameter of the Web server startup command is not valid. It can also be the result of an internal error which occurs when the Web server attempts to obtain a shared memory key and cannot get access to files in the /tmp directory.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Ensure that the WLM subsystem name specified on the -SN parameter is correct and that the Web server has read and write access to the /tmp directory.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0460E Creation of hash table failed, errno=*return_code*, **unable to cache**

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0461E Error trying to obtain well known key, errno=*return_code*

Explanation: This error can occur if the WLM subsystem name specified on the -SN parameter of the Web server startup command is not valid. It can also be the result of an internal error which occurs when the Web server attempts to obtain a shared memory key and cannot get access to the /tmp directory.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Ensure that the WLM subsystem name specified on the -SN parameter is correct.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0462E Failed getting access to message queue, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0463E Receive on message queue failed, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0464E Removal of message queue failed, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0465E Send on message queue failed, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0466E Can't create well known file: *file_name*, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0467E Failure initializing semaphore, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0468E Error trying to obtain well known key, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0469E Error getting semaphore, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0470E Error removing semaphore, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0471E Failed on attach of shared memory, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0472E Failed on detach of shared memory, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0473E Failed getting access to shared memory, errno=*return_code*

Explanation: An internal error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0474E Subsystem name *WLM_subsystem_name* is already in use, errno=*return_code*

Explanation: Another Web server with the same WLM subsystem name is already started.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Ensure that the WLM subsystem name specified on the -SN parameter of the Web server startup command is unique.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0475E Can't delete temporary file: *file_name*, errno=*return_code*

Explanation: An internal Web server error occurred.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0476E Cannot open the PICS error log: *file_name*, errno=*return_code*, errno2=*reason_code*

Explanation: An error occurred when the Web server attempted to open the PICS error log.

The z/OS UNIX System Services *return_code* and *reason_code* provide information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0477E Warning: PICS config file *file_name* could not be loaded. errno=*return_code*, errno2=*reason_code*

Explanation: An error occurred when the Web server attempted to load the PICS configuration file.

The z/OS UNIX System Services *return_code* and *reason_code* provide information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information

on the Web, see "Support services and resources" on page 604.

IMW0478E Unknown site option *name* **on a PICS site.**

Explanation: An error occurred when the Web server attempted to find the option in this message.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0479E PICS label for file *file_name* **specified outside a LabelsFor paragraph.**

Explanation: The Web server detected a PICS configuration error.

User Response: Correct the configuration error.

IMW0480E SYNTAX ERROR on line *number* **of protection file** *file_name***. Token:** *string*

Explanation: The Web server detected a syntax error.

User Response: Correct the syntax error.

IMW0481E SYNTAX ERROR on line *number* **of inline Protection directive** *name* **Token:** *string*

Explanation: A syntax error was detected.

User Response: Correct the syntax error.

IMW0482E Exec failed: *string***:** *string*

Explanation: An internal Web server error occurred.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0483E Not found. File or directory does not exist or is read-protected:*name*

Explanation: The Web server could not access the file or directory in this message.

User Response: Ensure that the file or directory exists and that the name was specified correctly.

IMW0484E Input timer expired while serving client *client_name* **for request** *request*

Explanation: The value set by the Web server's InputTimeout configuration directive has expired while waiting on a browser to submit another request.

User Response: Set the InputTimeout value higher to improve overhead in responding to a series of requests from a browser or lower to free up sessions more quickly.

IMW0485E Input timer expired while waiting on client *client_name*

Explanation: The value set by the Web server's InputTimeout configuration directive has expired while waiting on a browser to submit another request.

User Response: Set the InputTimeout value higher to improve overhead in responding to a series of requests from a browser or lower to free up sessions more quickly.

IMW0486E Persist timer expired while serving client *client_name* **for request** *request*

Explanation: The value set by the Web server's PersistTimeout configuration directive has expired while waiting on a browser to submit another request.

User Response: Set the PersistTimeout value higher to improve overhead in responding to a series of requests from a browser or lower to free up sessions more quickly.

IMW0487E Persist timer expired while waiting on client *client_name*

Explanation: The value set by the Web server's PersistTimeout configuration directive has expired while waiting on a browser to submit another request.

User Response: Set the PersistTimeout value higher to improve overhead in responding to a series of requests from a browser or lower to free up sessions more quickly.

IMW0488E mkDPIregister() failed for *string***.**

Explanation: An internal Web server error occurred.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0489E DPI cannot send *name* **MIB registration packet. rc=***return_code* **errno=***z/OS_UNIX_return_code***, errno2=** *z/OS_UNIX_reason_code***.**

Explanation: The SNMP subagent could not send the registration packet for the Web server MIB to SNMP. SNMP support will not operate.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Make sure the SNMP agent is running and the SNMP agent is DPI capable.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0490E DPI received no reply to *name* **MIB registration. rc=***return_code* **errno=***z/OS_UNIX_return_code***, errno2=** *z/OS_UNIX_reason_code***.**

Explanation: The SNMP subagent did not receive a response to the registration packet for the Web server MIB from the SNMP agent. SNMP support will not operate.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: Make sure the SNMP agent is running and the SNMP agent is DPI capable.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0491E Unknown INTERNAL function *function_name***.**

Explanation: An internal Web server error occurred.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0492E Can't find home directory: *directory*

Explanation: An error occurred when the Web server attempted to find the directory in this message.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0495E An error occurred while allocating directivebuf, size = *number*

Explanation: An internal Web server error occurred.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0496E [an error occurred while processing this servlet tag] Name=*name* **Code=***return_code* **Codebase=***version* **500:***string*

Explanation: The Web server detected a servlet syntax error.

User Response: Correct the syntax error.

IMW0497E [an error occurred while processing this servlet tag] Name=*name* **Code=***return_code* **Codebase=***version* **500: No details available**

Explanation: The Web server detected a servlet syntax error.

User Response: Correct the syntax error.

IMW0498E HTUid.... Must be MVS UserID, *string***,** *string* **or** *string***, not** *string*

Explanation: The Web server detected an error in the MVS UserID error.

User Response: Correct the error.

IMW0499E HTUid.... Must be MVS UserID, *string* **or** *string***, not** *string*

Explanation: The Web server detected an error in the MVS UserID error.

User Response: Correct the error.

IMW0500E GWAPI....Internal error-UNKNOWN server state.

Explanation: An internal Web server error occurred.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0501E Accept soft error: Errno: *return_code* **Errno2:** *reason_code* **Error:** *error*

Explanation: This message occurs when certain types of errors occur in the Web server's connection to a browser. These errors include ENOBUFS, EMFILE, ENFILE, ECONNABORTED, EIO, ENOMEM, ETIMEDOUT, and others. They exclude ECONNRESET, ECONNREFUSED, EINTR, EWOULDBLOCK and EINTRNODATA. If several errors occur in series without any successful I/O, this message is issued only for the first in the series, until at least one successful I/O operation is completed.

The z/OS UNIX System Services *return_code* and *reason_code* provide information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: This message can generally be ignored; however, if you see a lot of these messages, it can indicate a problem. If too many soft errors occur without any successful I/O, message IMW0502E is issued, and the Web server shuts down.

Note the information in the message and contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0502E Accept() soft error count excessive...shutting down

Explanation: This message occurs when too many soft errors of a certain type occur in the Web server's connection to a browser. See message IMW0501E for a list of those errors. If too many soft errors occur without any successful I/O, this message is issued, and the Web server shuts down.

User Response: See message IMW0501E for error codes and additional information.

IMW0503E Accept on master socket failed...shutting down: Errno *return_code* **Errno2** *reason_code* **Error** *description***.**

Explanation: This message occurs when the Web server cannot open a connection to receive a request from a browser. APARs PQ34123 and PQ34823 (2/00) fixed this problem.

User Response: If you need to contact the IBM Software Support Center for assistance, note the z/OS UNIX System Services *return_code*, *reason_code*, and *description* in this message. For support information on the Web, see "Support services and resources" on page 604.

IMW0504W SYNTAX ERROR: LDAPInfo *label_name* **subdirective** *name* **contains an invalid value of** *value*

Explanation: A syntax error was detected.

User Response: Correct the syntax error.

IMW0505W SYNTAX ERROR: LDAPInfo *label_name* **contains an unrecognized subdirective** *name*

Explanation: A syntax error was detected.

User Response: Correct the syntax error.

IMW0506W Could not establish communications with *number* **LDAP servers defined in the configuration.**

Explanation: One or more LDAP servers are defined in Web server LDAPInfo directives, but the Web server is not able to communicate with them.

User Response: Make sure that the LDAP servers are running. Check the LDAPInfo directives for correctness, especially the IP name or address specified in the Host subdirective. Other subdirectives in the LDAPInfo directive that could cause a connection failure are: Transport, Port, ServerAuthType, ServerDN, ServerPasswordStashFile, KeyFileName, KeyFilePasswordStashFile, and KeyLabel.

IMW0507W LDAP: unable to load library *'module_name'***; LDAP disabled.**

Explanation: LDAP is not able to load the LDAP API module name in the message during LDAP initialization. The Web server cannot communicate with LDAP servers until the problem is resolved.

User Response: Consult with your system administrator to ensure that LDAP support is properly set up.

IMW0508W LDAP: unable to find function *'function_name'***;** *'error description'***; continuing ...**

Explanation: LDAP initialization is not able to determine the address of the function named in the message. The Web server may not be able to communicate with LDAP servers if the function is required for LDAP support. If the function is not used in your LDAP implementation, communication may not be affected and processing will continue.

User Response: You may need to consult with your system administrator to ensure that LDAP support is properly set up.

IMW0509W LDAP: multiple LDAPInfo definitions for label *'label_name'***.**

Explanation: A configuration error was detected.

User Response: Correct the configuration error.

IMW0510W LDAP: missing Host field from LDAPInfo *'label_name'***.**

Explanation: A configuration error was detected.

User Response: Correct the configuration error.

IMW0511W LDAP: reference to an undefined LDAPInfo label: *'label_name'***.**

Explanation: A configuration error was detected.

User Response: Correct the configuration error.

IMW0512W LDAP: unable to find user *'name'* **with search filter** *'filter_string'***; rtn_code=***rtn_code_value***:***rtn_code_text***.**

Explanation: The LDAP server cannot authenticate the user in the message because the user name cannot be found. This message also includes the LDAP search filter that the Web server used, the return code that occurred, and a text description of the return code.

User Response: If the user should be authenticated, ensure that the user is in the LDAP server database. Also, check the UserSearchBase, UserNameFilter, and

other subdirectives on the Web server LDAP directive for correctness.

IMW0513W LDAP: *'name'* is not a unique username; *number* entries matched.

Explanation: More than one match was returned to the Web server when the LDAP server attempted to authenticate a user. Since authentication must identify each user uniquely, the authentication fails.

User Response: Remove any duplicate entries from the LDAP database.

IMW0514W LDAP: failure to find unique entry for *'name'* using LDAPInfo *'label_name'*.

Explanation: More than one match was returned to the Web server when the LDAP server attempted to authenticate a user. Since authentication must identify each user uniquely, the authentication fails.

User Response: Remove any duplicate entries from the LDAP database.

IMW0515W LDAP: unable to get first entry.

Explanation: An internal error occurred. The LDAP authentication routine performed a search of the LDAP database and returned one entry, but the LDAP_first_entry function could not locate the entry in memory.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

If IBM Support requests a trace, run the Web server with trace and ldaptrace active to get more information on the returned entry. To turn on tracing, specify -vv and either -debug ldap or -debug ldapv in the startup command. For information on the use of -vv with -debug, see "Tracing options" on page 405.

IMW0516W LDAP: unable to get distinquished name.

Explanation: An internal error occurred. The LDAP authentication routine performed a search of the LDAP database and returned one entry. The ldap_first_entry function located the entry in memory, but the ldap_get_dn function could not find a distinguished name in the entry.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

If IBM Support requests a trace, run the Web server with trace and ldaptrace active to get more information on the returned entry. To turn on tracing, specify -vv and either -debug ldap or -debug ldapv in the startup command. For information on the use of -vv with -debug, see "Tracing options" on page 405.

IMW0517W LDAP: basic authentication failure for *distinguished_name*: *error_text*.

Explanation: LDAP was called to authenticate a request's user, but a corresponding entry was not found in the LDAP database.

User Response: Ensure that the distinguished name in the message is expected, and update the LDAP database if needed. The error text in the message provides additional information about the cause of the failure.

IMW0518W LDAP: authentication failure: a valid certificate is required.

Explanation: The Web server configuration file specified `ClientAuthType Cert` for the LDAP server definition associated with the protection setup for this request. This indicates that an SSL certificate is required, but there is no SSL certificate.

User Response: Ensure that the client has a valid SSL certificate installed in the browser, then retry the request. If the default SSL port of 443 is not used, ensure that the client specifies the SSL port number in the request; this is the port number specified on the SSLPort directive in the Web server configuration file. This failure can also occur if the client makes the request using a non-SSL session.

IMW0519W LDAP: failure searching for groups of LDAPInfo *label*: *error_text*.

Explanation: The Web server configuration file specified `groupfile "%%LDAP..."` in the LDAPInfo directive named in the label, but the attempt to read the entries in this groupfile was not successful.

User Response: Check the LDAPInfo directive's GroupSearchBase subdirective, and ensure that groups have been properly set up in the LDAP database. The error text in the message may consist of a single digit. If the digit is "1", the Web server was unable to obtain a connection to the LDAP server. If the digit is "2", the LDAP search for group members did not encounter an error, but the LDAP search did not find any members. If the error text is longer than a single digit, then the error text comes from a call to the ldap_err2string() function. This function is documented in the LDAP server documentation.

To access this documentation on the Web, go to the z/OS Book Server Web site at URL:

```
http://www.ibm.com/servers/eserver
/zseries/zos/bkserv/
```

Note: The URL appears on two lines for printing purposes. Enter the URL on one line in your browser.

IMW0520W LDAP: failure finding attribute *attribute* **with filter** *group_search_filter*:*error_text*.

Explanation: An error occurred when the Web server attempted to find the attribute in the message. This attribute is specified on the GroupMemberAttrs subdirective in an LDAPInfo directive. The attempt to find the attribute using the group search filter in the message was not successful.

User Response: Check the LDAPInfo directive's GroupSearchBase and GroupNameFilter subdirectives to ensure they are specified correctly in the configuration file. Note that you cannot use the %v# format in the GroupNameFilter. Ensure that the group search filter is correct in the configuration file. The error text in the message provides additional information about the cause of the failure.

IMW0521W LDAP: no matches for filter *group_search_filter*.

Explanation: The LDAP server did not find any entries using the group search filter in the message.

User Response: Check the Web server LDAPInfo directive's GroupSearchBase and GroupNameFilter subdirectives to ensure they are specified correctly in the configuration file. Note that you cannot use the %v# format in the GroupNameFilter. The error text provides additional information about the cause of the failure.

Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

If IBM Support requests a trace, run the Web server with trace and ldaptrace active to see the actual results of the search. To turn on tracing, specify -vv and either -debug ldap or -debug ldapv in the startup command. For information on the use of -vv with -debug, see "Tracing options" on page 405.

IMW0522W LDAP: no values for attribute *attribute*.

Explanation: While processing the LDAP group search results, the Web server could not find any value for the attribute named in the message.

User Response: Ensure that the GroupMemberAttrs subdirective has been coded correctly and that the LDAP entry for this group contains the attributes named in the message.

IMW0523W LDAP: unable to connect to *hostname* **at port** *port_number*.

Explanation: The Web server was not able to connect to the LDAP server named in the message.

User Response: Ensure that the LDAP server is running and serving requests.Check the LDAPInfo configuration directive's Host, Port, ServerAuthType, ServerDN, and ServerPasswordStashFile subdirectives, and the value in the ServerPasswordStashFile.

IMW0524W LDAP: unable to set *option* **option for** *label*:*error_text*

Explanation: The LDAP server with which the Web server is communicating is not able to support the option specified, or the values are outside the allowed range.

User Response: Ensure that the values of the Version and SearchTimeOut subdirectives are correct. Check for the following options and errors:

LDAP_OPT_REFERALS
The Web server was not able to set the option to find referrals to other LDAP servers.

LDAP_OPT_PROTOCOL_VERSION
The Web server was not able to set the LDAP protocol version.

LDAP_OPT_TIMELIMIT
The Web server was not able to set the SearchTimeOut.

LDAP_SSL_ALREADY_INIT
The Web server is unable to set a Secure Sockets Layer (SSL) option. The Web server can have problems when connecting over SSL to both clients and LDAP servers. Apply LDAP APAR OW51259 to resolve the problem.

The error text in the message can provide additional information about the cause of the problem.

IMW0525W LDAP: SSL failure: *error_text*.

Explanation: TransportSSL is specified in the Web server configuration file, but the Web server was not able to connect to the LDAP server using an SSL connection.

User Response: Ensure that the LDAP server will accept SSL connections, and that the value in the Port subdirective is correct for the LDAP server. The error text in the message may provide additional information about the cause of the failure.

IMW0526W LDAP: unable to authenticate for LDAPInfo *label*: *error_text*.

Explanation: The Web server was not able to connect to the LDAP server named in the message to perform authentication services.

User Response: The error text in this message and error messages issued prior to this message provide additional information about why the LDAP authentication services are not available.

IMW0527W LDAP: buffer overflow on argument *argument_number* **with filter** *filter***.**

Explanation: The Web server does not have enough storage to set up the search filter.

User Response: Check the%v# fields specified in the search filter and the corresponding data from the SSL certificate. Shorten the filter if possible.

IMW0528W LDAP: invalid information handle.

Explanation: An internal error occurred. The Web server lost the pointer from the Web server Protection setup to the LDAPInfo definition.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0529W LDAP: unable to load IKEYMAN library *library***.**

Explanation: The Web server was not able to load the IBM Key Management Utility (IKEYMAN) DLL, keyman.dll.

User Response: Ensure that the LIBPATH environment variable in the httpd.envvars file includes the library where the keyman.dll is stored. Check the permission bit settings to ensure that keyman.dll is executable.

IMW0530W LDAP: failure to convert keyfile *keyfile_name* **to the appropriate format.**

Explanation: The Web server was not able to convert the key file in this message to a usable format. The keyfile is not readable, or its internal format is not acceptable to the SSL key management functions.The key file name is defined on a KeyFileName subdirective of the LDAPInfo directive.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0532E ReadContent.. Premature End Of File on socket *number***- reason** *number*

Explanation: This message is issued when the Web server detects an error in the content-length information sent by a client. This error can occur if the Web server encounters an end of input from the socket before reading the number of bytes indicated by the content-range header; if valid information is not found on the next chunk; or if the Web server cannot find the required end-of-data indicator. Possible reasons are:

- The input buffer is empty, and the Web server expects more data. However, the Web server is unable to read any data.
- The Web server successfully read some data into the input buffer, read all the bytes indicated by the content-range header, and the input is chunked. However, the Web server was not able to find an end-of-data indicator.
- The Web server read all the bytes indicated by the content-range header, and the input is chunked. However, the Web server attempted to read the length information for the next chunk and the end-of-data indicator, and was unable to read them.
- The Web server read all the bytes indicated by the content-range header and the input is chunked. The Web server was able to read the length information for the next chunk and the length is zero. However, the Web server was unable to read the end-of-data indicator.

The incorrect content-range information was probably generated by a client applet. The error can also be caused by an I/O error on a socket.

User Response: Correct the applet which generated the incorrect content-range information.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0533E ReadContent.. Invalid chunk found when reading on socket *number*

Explanation: This message is issued when the Web server detects an error in the content-length information sent by a client. The Web server read all the bytes indicated by the content-range header, and the input is chunked. However, when the Web server attempted to read an end-of-data indicator, it received more data.

The incorrect content-range information was probably generated by a client applet. The error can also be caused by an I/O error on a socket.

User Response: Correct the applet which generated the incorrect content-range information.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0534E ReadContent.. Invalid final chunk found when reading on socket *number*

Explanation: This message is issued when the Web server detects an error in the content-length information sent by a client. The Web server read all the bytes indicated by the content-range header and the input is chunked. The Web server was able to read the length information for the next chunk and the length is zero. However, when the Web server attempted to read an end-of-data indicator, it received more data.

The incorrect content-range information was probably

generated by a client applet. The error can also be caused by an I/O error on a socket.

User Response: Correct the applet which generated the incorrect content-range information.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0535E ReadContent.. Chunk too big - error

Explanation: This message is issued when the Web server detects an error in the content-length information sent by a client. The Web server read all the bytes indicated by the content-range header and the input is chunked. The Web server was able to read the length information for the next chunk, but the length is not valid.

The incorrect content-range information was probably generated by a client applet. The error can also be caused by an I/O error on a socket.

User Response: Correct the applet which generated the incorrect content-range information.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0537E Storage request failed: unable to allocate *number* **bytes, errno =** *return_code***.**

Explanation: A request for storage failed.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0539E Server root entered on the httpd -sr command is not valid: *server_root*

Explanation: You entered a server root for the Web server that is not valid.

User Response: Correct the server root and reenter.

IMW0540E DIRECTORY BROWSE FAILED

Explanation: The listing of a directory that the client requested could not be served. Either the client is not permitted to browse the directory, or the directory does not exist.

User Response: This may be a permission bit problem. Review the permission bits of the directory being requested to see if the correct settings are defined.

IMW0541E OPEN FAILED

Explanation: The client requested a file that could not be opened. Either the client is not permitted to view the file, or some other open failure occurred.

User Response: This may be a permission bit problem. Review the permission bits of the directory being requested to see if the correct settings are defined.

IMW0542E BAD REQUEST

Explanation: The client request is not valid. The requested resource could not be served.

User Response: None.

IMW0543E PROXY LOAD FAILED

Explanation: The Web server attempted to proxy a client request, but an error occurred when attempting to receive the output from the content server.

User Response: None.

IMW0544E SCRIPT REQUEST NOT VALID

Explanation: A client request to execute a CGI program failed. Either no CGI name was specified, or the CGI could not be found.

User Response: None.

IMW0545E SCRIPT NOT EXECUTABLE

Explanation: The CGI progam requested by the client could not be executed.

User Response: Possible causes of this problem:

- The CGI program may not be marked executable.
- This may be a permission bit problem. Review the permission bits of the CGI program being requested to see if the correct settings are defined.

IMW0546E SCRIPT START FAILED

Explanation: The CGI program requested by the client could not be started. Either the pipe to the CGI process could not be created, or the spawn of the process failed.

User Response: Review entries in the error log to determine the cause of the failure.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0547E SCRIPT I/O ERROR

Explanation: A client request to execute a CGI program failed. An error occurred when the Web server attempted to open the pipe to read the output from the CGI.

User Response: Review entries in the error log to determine the cause of the failure.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW0548E PRECONDITION FAILED

Explanation: A client request condition contained an IF-MATCH or IF-NONE-MATCH header, but the resource being requested failed to meet the criteria of the header.

User Response: None.

IMW0549E PROXY HOST NOT FOUND

Explanation: The Web server attempted to proxy a client request, but the host name in the request is not defined.

User Response: None.

IMW0550E PROXY HOST CONNECT FAILED

Explanation: The Web server attempted to proxy a client request with a valid host name, but the connection to the host failed.

User Response: None.

IMW0551E WLM ENVIRONMENT UNAVAILABLE

Explanation: The request matched a Web server ApplEnv definition, but the transfer of work to a queue server was not successful. The request may be processed in the queue manager.

User Response: None.

IMW0552E BAD RANGE

Explanation: The PUT request sent by the client to the Web server contained a Content-Range header that did not have a valid range. The request is not processed.

User Response: None.

IMW0553E EXPECTATION FAILED

Explanation: A client sent a request to the Web server that contained an Expect header, but the header did not contain a value that the Web server recognized. The request is rejected.

User Response: None.

IMW0555E Value of ServerRoot directive is NULL

Explanation: A null value was specified on the ServerRoot directive in the configuration file.

User Response: Correct the configuration file.

IMW0556E Unable to determine HOSTNAME from value in the httpd.conf file.

Explanation: The value specified for the Hostname directive in the Web server configuration file was not found in the host name table.

User Response: Ensure that the value for the Hostname directive is correct, and that it has a matching host name table entry.

IMW0557E Failed getting access to message queue.

Explanation: Unable to obtain the message queue identifier associated with the key. Another message queue may already be using the key.

User Response: Contact the IBM software support center.

IMW0558E *GWAPI service:* **GWAPI handle is not valid; reason=***internal reason number***; handle=***hexadecimal value***.**

Explanation: The GWAPI service determined that it had been passed an invalid handle. The reason indicates which of several internal validity checks failed. This reason number is only useful to IBM.

User Response: The handle passed to the GWAPI program was corrupted before the GWAPI program called the GWAPI service function. Correct the GWAPI program in order to pass a valid handle.

IMW0559E *GWAPI service:* **GWAPI service was called, but the calling program was not a GWAPI; handle=***hexadecimal value***; t_private=***address of thread's private area in hexadecimal***.**

Explanation: The GWAPI service determined that the program which called it was not invoked through the normal GWAPI exit process.

User Response: Correct the GWAPI program to invoke the GWAPI services only from a program which has been invoked through the normal GWAPI exit process.

IMW0560E *GWAPI service:* **GWAPI program is already in a callback; handle=***hexadecimal value***; t_private=***address of thread's private area in hexadecimal***.**

Explanation: The GWAPI service determined that the program which called it was already in a GWAPI service.

User Response: Correct the GWAPI program to avoid creating multiple threads that invoke the GWAPI services simultaneously.

IMW0561E *GWAPI service:* **GWAPI return code argument is null.**

Explanation: The GWAPI service determined that the program which called it did not pass the address for the return code.

User Response: Correct the GWAPI program to pass the address of a return code when calling the GWAPI services.

IMW0562W **LDAP: unable to find user** *'name'* **with search filter** *'filter_string'***; rtn_code=***rtn_code_value***: none found.**

Explanation: The LDAP server cannot authenticate the user in the message because the user name cannot be found. However, no error condition was raised. This message also includes the LDAP search filter that the Web server used and the return code that occurred.

User Response: If the user should be authenticated, ensure that the user is in the LDAP server database. Also, check the UserSearchBase, UserNameFilter, and other subdirectives on the Web server LDAP directive for correctness.

IMW0563E **WorkQueue...** *WLM function_name* **failed; errno=***return_code***; errno2=***reason_code***; errmsg:** *explanation of errno*

Explanation: The Web server attempted a WLM operation, such as __server_pwu or __server_init, in connection with either scalable mode operations or the Fast Response Cache Accelerator. An error was encountered. The *WLM function_name* indicates the WLM function attempted.

The z/OS UNIX System Services *return_code* and *reason_code* provide information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If you receive this message while running in scalable server mode, verify that the:

- State of the WLM application environment is *available*.
- Application Environment name is coded correctly. This name is coded in the –AE parameter of the EXEC statement in the IMWIWM procedure and in the WLM policy. You can also code this name on the ApplEnv, ApplEnvConfig, and FRCAWLMParms directives.
- Subsystem name is coded correctly. This name is coded in the –SN parameter of the EXEC statement in the IMWIWM procedure and in the WLM policy. You can also code this name in the FRCAWLMParms directive.

If you receive this message while running in standalone mode, verify that the:

- WLMSN directives and Enclave directives are configured properly.
- The transaction class parameter of the Enclave directive is eight characters or less.
- The user ID used in the authentication step for the request is valid.

IMW0564E **CGI error: could not retrieve Web server environment variables.**

Explanation: The Web server was unable to generate the list of environment variables in order to pass the list to the CGI.

User Response: Check httpd.conf to verify that the setting for the DefaultFsCp directive is correct.

IMW0565E **GWAPI: could not retrieve Web server environment variables.**

Explanation: The Web server was unable to generate the list of environment variables in order to pass the list to the GWAPI.

User Response: Check httpd.conf to verify that the setting for the DefaultFsCp directive is correct.

IMW0566E **No SSLCipherSpecs are in use. SSL security is not available.**

Explanation: You did not code any SSL cipher specifications on the SSLCipherSpecs directive in the configuration file.

User Response: Code at least one SSL cipher specification on the SSLCipherSpecs directive.

IMW0567E **SSL initialization: return code from skit_initialize=***rtn_code_value*

Explanation: The Web server found a bad return code and could not initialize SSL.

In addition, the Web server returns a –1 when it cannot find the stash file, or if it does not have access to the file. Another Web server message accompanies this message to clarify the problem.

User Response: Review the description of the return code and correct the problem. For descriptions of the

return codes, see the *z/OS System SSL Programming Guide and Reference*.

To access this book on the Web, go to the z/OS Book Server Web site at URL:

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

Note: The URL appears on two lines for printing purposes. Enter the URL in your browser as one line.

You can also find return codes listed with their descriptions in the directory `/usr/lpp/gskss/include/gskssl.h`.

For the stash file problem, review the accompanying message and correct the problem.

IMW0568E Unable to open password stash file: *stash_file_name*.

Explanation: The Web server was unable to open the password stash file to initialize SSL. SSL does not function.

User Response: The password stash file must exist and must meet the following criteria:

- Have the same name as the key database file, except that the file extension is `.sth`.
- Be in the same directory as the key database file.
- Be readable by the Web server.

Example:

If the key database is named `key.kdb` and it is in the `/u/xyz` directory, then the stash file would be named `key.sth` and would also be in the `/u/xyz` directory. Define the permission bits so that the Web server can have read access to the file.

If the file does not exist, create it using the System SSL gskkyman utility.

If the stash file meets the preceding criteria, and the Web server still cannot open the file, the file could be corrupted. If corrupted, you must create the file again.

IMW0569E Cannot bind and listen on port *port_number*.

Explanation: The Web server cannot initialize SSL.

User Response: A previous message in the error log explains why SSL did not initialize. Correct the problem. If you do not use SSL, you can turn SSL off one of two ways:

- Code `SSLMode Off` in the configuration file.
- Code `-sslmode off` on the PARM keyword of the EXEC statement in the Web server procedure.

IMW0570W WARNING: Wildcard ApplEnv name is allowed but not recommended.

Explanation: You used a wildcard (*) in the application environment name (AEName) field of the ApplEnv directive. This value is supported, but not recommended.

User Response: Explicitly specify the AEname in all instances of the ApplEnv directive.

IMW0571E The minimum number of threads needed for the current configuration is *minimum number*. The value of the MaxActiveThreads directive is *MaxActiveThreads number* —— SERVER STOPPING!!!

Explanation: The number of threads that you allocated on MaxActiveThreads is less than the number of threads that the Web server needs in order to execute requests.

User Response: Use the following scale to determine the minimum number of threads that are needed for your configuration. Then set MaxActiveThreads to at least the minimum. Assume that the number of threads needed for the Web server to initialize is zero.

- If the SSLMode directive is set to the value ON, you need at least one thread to process SSL requests. Add one to the minimum.
- If the NormalMode directive is set to the value ON, you need as least one thread to process non-SSL requests. Add one to the minimum.
- If the Web server is executing in scalable mode, you need at least two threads. Add two to the minimum.

If SSLMode is set to the value ON, NormalMode is set to the value ON, and you execute in scalable mode, at a minimum set MaxActiveThreads to 4. However, the program default setting for MaxActiveThreads is 40. For normal execution of the Web server, there is really no need for you to set MaxActiveThreads anywhere near the minimum number of threads required.

IMW0572E Value for QoS is not valid: *string*

Explanation: The value that you coded on the QOS directive is incorrect. It is not one of the valid values.

User Response: Modify the QOS directive parameter to either HOST, URI, or HOSTURI.

IMW2000E-2026E: Proxy Server Messages

IMW2000E Proxy Error: Host name not recognized or host not found.

Explanation: This error is returned when an incorrect or unrecognizable host name is specified. The proxy server has tried to parse the host name and found the name to be not valid (for example, xyz.ibm.com).

User Response: Verify that the name of the destination server is correct.

IMW2002E Proxy Error: Unable to connect to remote host or host not responding

Explanation: This error is returned when the proxy server is unable to establish a socket connection with the remote host. The remote host is down or for some reason unavailable, as a network connection cannot be established.

User Response: Try again to establish the connection. The destination server may be temporarily unavailable.

IMW2003E Proxy Error: SSL Handshake with upstream server failed.

Explanation: This error occurs when the proxy server is unable to establish an SSL connection. To generate this error, cause a handshake failure between the secure server and secure proxy server. One way to do this is to configure disjointed certificates, that is, each side has a certificate that the other does not trust.

User Response: Verify that the certificates between the two servers are able to authenticate.

IMW2004E Proxy Error: Host connected but unable to send request. Please retry.

Explanation: This error occurs when the proxy server established a socket connection with the destination server, but then the destination server immediately closed the connection. This might be caused by a degenerate Web server. To generate this error, write a simple application, or modify HOWL, to accept the socket connection and then immediately drop the connection.

User Response: Send the request again. The destination server may be temporarily unavailable.

IMW2005E Proxy Error: Host connected but unable to forward request. Please retry.

Explanation: This error occurs when the proxy server has sent the request headers, but the connection with the destination server is lost before the proxy server can send the content body. This scenario is easy to create for test purposes. Try sending the request header followed by a huge content body (for example 64KB) on a POST request, and then closing the connection after the request headers have been read, but before the content body has been received by the destination server. Use a debugger to get the timing right.

User Response: Resend the request. The remote server may be temporarily unavailable.

IMW2006E Proxy Error: Remote host did not send any data.

Explanation: The destination server did not respond to the proxy server's request. In most cases, the destination server or network connection went down before the server could respond.

User Response: Retry the request.

IMW2007E Invalid forced expiry time spec: *string*.

Explanation: The syntax for the CacheMinHold directive in the javelin configuration file is incorrect or was not specified.

User Response: Correct the error. See the IBM Web Traffic Express for Multiplatforms User's Guide V1.0 for the correct syntax for the CacheMinHold directive.

IMW2008E Javelin..... sox.conf could not be opened.

Explanation: Unable to open the socks configuration file (socks.conf). This file is in the /etc directory.

User Response: Ensure that the file exists and check the file permissions.

IMW2009E PICS Rule starting at line *number* **could not be parsed.**

Explanation: There is an error in the PICS Rule.

User Response: Check the syntax of the PICS Rule in javelin configuration file (for example, javelin.conf).

IMW2010E PICS Rule starting at line *number* **could not be parsed. The error was detected near the text** *string*.

Explanation: There is an error in the PICS Rule.

User Response: Check the syntax of the PICS Rule in the javelin.conf configuration file.

IMW2012E Cannot open configuration file: *filename*.

Explanation: The Web server was unable to open the javelin configuration file.

User Response: Check for the existence and permissions of the javelin configuration file.

IMW2013E Unknown Javelin directive *directive*.

Explanation: The specified directive from the javelin configuration file is unrecognized.

User Response: Remove this directive from the javelin configuration file.

IMW2014E Too many arguments on one line in configuration file, max: *number*

Explanation: There were too many arguments on one line.

User Response: Change the number of arguments on the line for the directive.

IMW2015E Insufficient parameters for directive: *name*

Explanation: There were not enough arguments on the line.

User Response: Change the number of arguments on the line for the directive.

IMW2016E Invalid Number of subcaches: *string***.**

Explanation: A negative number was specified for the ProxyNumTables directive in the javelin configuration file.

User Response: The number of subcaches must be equal to or larger than zero.

IMW2017E PICS rule group file couldn't be read.

Explanation: The Web server could not read the referenced PICS rule containing the group file subdirective. Either the file does not exist or it contains unreadable information.

User Response: Change the referenced PICS rule to point to a valid file or ensure the file contains valid information.

IMW2018E PICS rule applies-to couldn't be read.

Explanation: The referenced PICS rule contains an applies-to subdirective that is not valid.

User Response: Correct the PICS rule by entering a valid applies-to subdirective.

IMW2019E Cannot run cache agent because proxy server is not specified.

Explanation: An error occurred when the Web server attempted to run the cache agent. This error can occur when the Proxy directive is not specified in the javelin configuration file.

User Response: Specify the host and port for the proxy server on the Proxy directive in the Web server javelin configuration file.

IMW2020E NumClients too large. Downgraded number of cache refresh clients to 100.

Explanation: NumClients exceeded 100 and was decreased to the limit of 100 threads. This is to minimize network use and stress on the content servers that will be contacted.

User Response: Decrease the number of threads for the cache agent. See the NumClients directive in the Web server javelin configuration file.

IMW2021E Could not open the cache agent configuration file *filename***.**

Explanation: Unable to open the javelin configuration file.

User Response: Check the existence and the permissions of the javelin configuration file.

IMW2022E Not able to load files specified in cache log, check proxy server configuration.

Explanation: The cache agent was unable to load files that were specified in the cache access log. This can be caused by lack of a cache access log, inability to locate the Web server configuration file, or having caching set off.

User Response: Check the configuration files and system for the three symptoms listed above and correct it, as appropriate.

IMW2023E The proxy server is not configured for caching, specify CachingOn.

Explanation: Caching is set off in the Web server configuration file.

User Response: Set the Web server Caching directive to on.

IMW2024E Proxy server did not specify CacheAccessLog directive.

Explanation: The Web server CacheAccessLog directive is not set

User Response: Set the CacheAccessLog in the Web server configuration file.

IMW2025E Could not open the proxy server's CacheAccessLog file *filename***.**

Explanation: Explanation: The proxy server's cache access log could not be opened.

User Response: Check that the CacheAccessLog directive is set. Check for the existence and permissions of the corresponding log file.

IMW2026E Forbidden - not allowed

Explanation: The user has issued a request which is being proxied to the origin server using reverse proxy. Access to the proxy server is protected at the proxy server, and the file being requested from the origin server is protected at the origin server. Protection at both the proxy server and the origin server at the same time is not supported in the reverse proxy environment.

User Response: To access the file using reverse proxy, remove the protection from either the proxy server or from the file at the origin server. If both access to the proxy server and access to the file at the origin server must be protected, then reverse proxy cannot be used to access the file. In that case, the file must be accessed using the proxy server as a normal proxy server and not a reverse proxy server.

IMW3501I-3546E: CONSOLE Messages

Explanation of process descriptor in messages

Several CONSOLE messages contain a *process_descriptor* variable, which includes *mode processid ServerIPA*:*ServerPort serverSN serverAE*:

mode
is one of:
- SA - the server started as a standalone process (not managed by WLM)
- QM - the server started as a WLM Queue Manager
- QS - the server started as a WLM Queue Server

processid
is the server's process identifier as assigned by z/OS UNIX System Services

ServerIPA
is the server's IP address, in the form n.n.n.n, where n ranges from 0 to 255. If *ServerIPA* is 0.0.0.0, then the server is not bound to any specific IP address (if neither the BindSpecific directive nor the -HN start parameter was used).

ServerPort
is the primary port number that the server is listening on.

serverSN
is the WLM subsystem name. If *mode* indicates SA, then *serverSN* will be *. Otherwise, it will reflect the value passed as the -SN start parameter.

serverAE
is the WLM Application Environment name. If *mode* indicates SA or QM, then *serverAE* will be *. Otherwise, it will reflect the value passed as the -AE start parameter.

Message descriptions

IMW3501I Config: Hostname: *hostname*, **Port:** *number*, **SSL Port:** *number*, **Server Root:** *serverroot*. *current_debug_settings current_smf_setting*

Explanation: Displays the server's hostname, normal port, and server root. The SSL port is specified if the server is running in secure mode. If the server is not running in secure mode, NONE is indicated for the SSL port. Debug setttings that are turned on are listed. The SMF recording setting is also displayed.

User Response: None.

IMW3502I **Stats: Threads running:** *number*, **Threads idle:** *number*, **Requests:** *number*, **Bytes rcvd:** *number*, **Bytes sent:** *number*, **Actv In Conns:** *number*, **Actv Out Conns:** *number*. **Connections since last SMF:** *number* **DNS Max:** *number*, **DNS Min:** *number*, **DNS Avg:** *number*, **Service Plugins Max:** *number*, **Service Plugins Min:** *number*, **Service Plugins Avg:** *number*, **CGI Max:** *number*, **CGI Min:** *number*, **CGI Avg:** *number*, **SSL Handshake Max:** *number*, **SSL Handshake Min:** *number*, **SSL Handshake Avg:** *number*, **Proxy Response Max:** *number*, **Proxy Response Min:** *number*, **Proxy Response Avg:** *number*, **Non-SSL Waiting Threads:** *number*, **SSL Waiting Threads:** *number*, **Async I/O Waiting Threads:** *number*, **Msg Queue Waiting Threads:** *number*.

Explanation: Displays activity statistics for the Web server.

For more information on these statistics, see "Web server activity statistics" on page 249.

User Response: None.

IMW3503I **Do not recognize option,** *parameter*, **on -d option.**

Explanation: You entered a parameter that is not valid on the -d option on the MODIFY command.

User Response: The message displays the valid options. Be sure you entered an acceptable option.

IMW3504I **Debug has been enabled for module,** *module_name*.

Explanation: You entered -debug *module_name* on the MODIFY command. Debugging for the named module has been enabled.

User Response: None.

IMW3505I **Debug has been enabled for all modules.**

Explanation: You entered -debug on the MODIFY command. Debugging has been enabled for all modules.

User Response: None.

IMW3506I **Unknown module name,** *module_name*, **specified on the -debug option.**

Explanation: You specified -debug *module_name* with an unknown *module_name*.

User Response: Correct the module name and try again. (Valid module names are listed in message IMW3523I.)

IMW3507I **Debug has been disabled for module,** *module_name* **.**

Explanation: You entered -nodebug *module_name* on the MODIFY command. Debugging for the named module has been disabled.

User Response: None.

IMW3508I **Debug has been disabled for all modules.**

Explanation: You entered -nodebug on the MODIFY command. Debugging for all modules has been disabled.

User Response: None.

IMW3509I **Unknown module name,** *unknown_module_name* **, specified on -nodebug option. Option is: -nodebug** *module_name*. **Turns of debugging for all modules or optionally specify one of the following:** *list of module names*

Explanation: The specified module name is not valid.

User Response: Correct the module name and try again. Valid module names are listed in the message.

IMW3510I **SMF recording has been enabled for record type,** *record_type* **.**

Explanation: You entered -smf *record_type* on the operator console MODIFY command. SMF recording for the named record type has been enabled.

User Response: None.

IMW3511I **SMF recording has been enabled for all record types.**

Explanation: You entered -smf on the operator console MODIFY command. SMF recording has been enabled.

User Response: None.

IMW3512I **Do not recognize option,** *record_type* **, on the -smf option.**

Explanation: You specified -smf *record_type* with an unknown *record_type*.

User Response: Correct the *record_type* and try again. Valid *record_types* are listed in the message.

IMW3513I SMF recording has been disabled for record type, *record_type*

Explanation: You entered -nosmf *record_type* on the operator console MODIFY command. SMF recording for the names record type has been disabled.

User Response: None.

IMW3514I SMF recording has been disabled for all record types.

Explanation: You entered -nosmf on the MODIFY command. SMF recording has been turned off.

User Response: None.

IMW3515I Do not recognize option, *record_type* **, on the -nosmf option.**

Explanation: You specified -nosmf *record_type* with an unknown *record_type*.

User Response: Correct the *record_type* and try again. (Valid *record_types* are listed in message IMW3526.)

IMW3516I Version: *version*

Explanation: You entered -version on the MODIFY command. The server version is displayed.

User Response: None.

IMW3517I First level tracing (-v) enabled.

Explanation: You entered -v on the MODIFY command. Verbose tracing has been enabled.

User Response: For guidance and additional information on Web server traces, contact the IBM Software Support Center. For support information on the Web, see "Support services and resources" on page 604.

IMW3518I Second level tracing (-vv) enabled.

Explanation: You entered -vv on the MODIFY command. Very Verbose tracing has been enabled.

User Response: For guidance and additional information on Web server traces, contact the IBM Software Support Center. For support information on the Web, see "Support services and resources" on page 604.

IMW3519I Third level tracing (-mtv) enabled.

Explanation: You entered -mtv on the MODIFY command. Much Too Verbose tracing has been enabled.

User Response: For guidance and additional information on Web server traces, contact the IBM Software Support Center. For support information on the Web, see "Support services and resources" on page 604.

IMW3520I Cache tracing (-vc) enabled.

Explanation: You entered -vc on the MODIFY command. Verbose cache tracing has been enabled.

User Response: For guidance and additional information on Web server traces, contact the IBM Software Support Center. For support information on the Web, see "Support services and resources" on page 604.

IMW3521I Do not recognize option, *option***, on the MODIFY command.**

Explanation: You entered an unknown option on the MODIFY command.

User Response: Message IMW3527I identifies valid options. Try the command again with a correct option.

IMW3522I No Help available for option, *option***.**

Explanation: You specified **-?** *option* on the MODIFY command. There is no help available for *option*. Help is available for debug (-? debug) and nodebug (-? nodebug), smf (-? smf), and nosmf (-? nosmf).

User Response: Enter **-??** to see more information.

IMW3523I Option is: -debug *module_name* **Enables debugging for all modules, or optionally specify one of the following:** *module_names listed*

Explanation: You should receive this message when -? debug is specified on the MODIFY command.

User Response: None.

IMW3524I Option is: -nodebug *module_name* **Disables debugging for all modules, or optionally specify one of the following: config, perf.**

Explanation: You should receive this message when -? nodebug is specified on the MODIFY command.

User Response: None.

IMW3525I Option is: -smf *SMF_record_type* **Enables all SMF recording, or optionally specify one of the following: config, perf.**

Explanation: You should receive this message when -? smf is specified on the operator console MODIFY command.

User Response: None.

IMW3526I **Option is: -nosmf** *SMF_record_type* **Disables all SMF recording, or optionally specify one of the following: config, perf.**

Explanation: You should receive this message when -? nosmf is specified on the operator console MODIFY command.

User Response: None.

IMW3527I **Options: -d config -d stats -debug** *module_name* **-nodebug** *module_name* **-restart -smf** *option* **-nosmf** *option* **-v -vv -mtv -vc -version -? -?? -? debug -? nodebug -? smf -? nosmf**

Explanation: You entered -? on the MODIFY command.

User Response: None.

IMW3528I **-d config : Server configuration -d stats : Server statistics -debug** *module_name* **: Enables trace for specified module(s) -debug : Traces all modules (maximum tracing) -? debug : Lists mod_names for -debug -nodebug** *module_name* **: Disables trace -? nodebug : Module names for -nodebug -restart : Restarts the Server -smf** *option* **: Enables SMF recording -? smf : Lists options for -smf -nosmf** *option* **: Disables SMF recording -? nosmf : Lists options for -nosmf -v : Verbose (first level tracing) -vv : Very Verbose (second level tracing) -mtv : Much Too Verbose (third level tracing) -vc : Verbose Cache (cache tracing) -version : Version of Server**

Explanation: You have entered -?? on the MODIFY command.

User Response: None.

IMW3529E **Error on console()..***string***. Internal error. This is a software error. Call IBM Software Support.**

Explanation: This is an internal error in the console support function.

User Response: Note the information in the message and contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3530E **Error on msg parameter to HTWTO(), msg=NULL, n_padding=***number***, flags=***hex_value*

Explanation: This is an internal error in the console support function.

User Response: Note the information in the message and contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3531E **Error on msg parameter to HTWTO(), strlen(msg)=0, msg=***number* **n_padding=***number***, flags=***hex_value*

Explanation: This is an internal error in the console support function.

User Response: Note the information in the message and contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3532E **Error on msg parameter to HTWTO(), strlen(msg)=0, n_padding=***number***, flags=***hex_value*

Explanation: This is an internal error in the console support function.

User Response: Note the information in the message and contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3533E **Error on n_padding parameter to HTWTO(), n_padding=***number***, HTWTOconsoleWIDTH=***number*

Explanation: This is an internal error in the console support function.

User Response: Note the information in the message and contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3534I **PID:** *processid* **SERVER STARTING**

Explanation: The server is initializing. *processid* is the server's numeric process identifier assigned by z/OS UNIX System Services. Message IMW3535E or IMW3536I should follow this message, indicating initialization failure or success.

User Response: None

IMW3535E **PID:** *processid* **SERVER INITIALIZATION FAILED**

Explanation: Server initialization started, but did not complete, due to one or more error conditions.

User Response: Review error messages that have been written to the job log (using DD STDERR) for a description of the error(s). *processid* is the server's process identifier. Correct the errors and restart the server.

IMW3536I *process_descriptor* **READY**

Explanation: The server has initialized successfully, and can process Internet requests. For more information, see "Explanation of process descriptor in messages" on page 661.

User Response: None.

IMW3537I *process_descriptor* **RESTARTING**

Explanation: The server has been requested to terminate and restart its processing (possibly an operator request to restart). The configuration file is read, which causes some (not all) functions to be recycled. During restart, new connections are not allowed, and existing connections are quiesced. IMW3538I or IMW3539E should follow this message, indicating a successful restart or failure. For more information, see "Explanation of process descriptor in messages" on page 661.

User Response: None.

IMW3538I *process_descriptor* **RESTART SUCCESSFUL**

Explanation: The server completed the restart successfully, and can process internet requests. For more information, see "Explanation of process descriptor in messages" on page 661.

User Response: None.

IMW3539E *process_descriptor* **RESTART FAILED**

Explanation: The server began to restart, but did not complete successfully, due to one or more errors. For more information, see "Explanation of process descriptor in messages" on page 661.

User Response: Review the error messages written to the job log (using DD STDERR) and correct the errors. Start the server again.

IMW3540I *process_descriptor* **STOPPING WORK**

Explanation: The server received a request to terminate processing (for example, an operator has issued a STOP command). The server quiesced its functions, new connections were not allowed, and existing connections are quiesced. Message IMW3541I should follow this message. For more information, see "Explanation of process descriptor in messages" on page 661.

User Response: None.

IMW3541I *process_descriptor* **TERMINATING NOW**

Explanation: Shutdown of the server is complete. The server has shut down all connections, or existing connections have timed out after 5 minutes. For more information, see "Explanation of process descriptor in messages" on page 661.

User Response: None.

IMW3542E *process_descriptor* **DUMPING**

Explanation: The server was notified of a program check (ABEND) during its processing and dumped to enable problem diagnosis. No more messages were issued, and the server ends immediately. For more information, see "Explanation of process descriptor in messages" on page 661.

User Response: Review the dump to diagnose the error. Start the server after correcting the problem.

IMW3543E MVS Console Support ERROR, Listen returns EINVAL.

Explanation: The Web server detected an error when writing a message to the MVS operator console. This is an internal software error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3544E MVS Console Support ERROR, Listen returns EFAULT.

Explanation: The Web server detected an error when writing a message to the MVS operator console. This is an internal software error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3545E MVS Console Support ERROR, Listen returns EMVSERR.

Explanation: The Web server detected an error when writing a message to the MVS operator console. This is an internal software error.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3546E MVS Console Support ERROR, retv=*code***, errno=***return_code*

Explanation: The Web server detected an error when writing a message to the MVS operator console.

code is the return code from a call to MVS console services.

The z/OS UNIX System Services *return_code* provides information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3701E-3730E: HTCounter Program Messages

IMW3701E HTCounter: Counter file not found, tried the following: *counterfile*

Explanation: A URL for this server specified the apicounter function available in the htcounter program. It tried to access a non-existent counter file.

User Response: Ask your Web administrator to create the counter file in the counters subdirectory of the server's root directory with an initial value in it (typically 0) and set permissions to allow the server write access.

IMW3702E HTCounter: strftime() call failed: TIMEFMT=*format_string***.**

Explanation: A URL for this server specified the datetime function available in the htcounter program. The call to strftime() to format the current date and time failed. The message shows the format option used on the call to strftime().

User Response: Note the information in the message and contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3703E HTCounter: software error: strftime() call failed: result of localtime_r() call is 0.

Explanation: A URL for this server specified the datetime function available in the htcounter program. The call to localtime_r() to determine the current date and time failed.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3704E HTCounter: software error: strftime() call failed: result of gm_time_r() call is 0.

Explanation: A URL for this server specified the datetime function available in the htcounter program. The call to gm_time_r() to determine the current date and time failed.

User Response: Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3705E HTCounter: strftime() call failed: (*form).Timebase=*string***.**

Explanation: A URL for this server specified the datetime function available in the htcounter program. This message indicates that an internal error occurred in the htcounter program.

User Response: Note the information in the message and contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3706E HTCounter: unable to open Counters/Fonts/FormsEtc.dat file..tried *filename***.**

Explanation: An error occurred while the htcounter program was attempting to load the fonts files. The file, FormsEtc.dat, should be located under the Counters/Fonts subdirectory in the server root subdirectory. The message indicates where the htcounter program attempted to load the file from.

User Response: This error could occur if there was an installation problem, or if this file was inadvertently moved or deleted. Search for the file, FormsEtc.dat, and restore it to the <ServerRoot>/Counters/Fonts subdirectory. It may be necessary to reinstall the server.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3707E HTCounter: too many fonts specified in *filename* **file.. maximum allowed=***number_allowed***., file asks for** *number_requested*

Explanation: The FormsEtc.dat file, named *filename*, was read and there is an error in the contents.

User Response: This error could occur if there was an installation problem, or if this file was inadvertently changed. Attempt to restore the original contents of the FormsEtc.dat file. It may be necessary to reinstall the server.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3708E HTCounter: unable to open Counters/Fonts/*font_filename* file..tried *full_font_filename*.

Explanation: An error occurred while the htcounter program was attempting to load one of the fonts files. The file, *font_filename*, should be located under the Counters/Fonts subdirectory in the server root subdirectory. The message indicates where the htcounter program attempted to load the file from.

User Response: This error could occur if there was an installation problem, or if this file was inadvertently moved or deleted. Search for the file, *font_filename*, and restore it to the subdirectory, <ServerRoot>/Counters/Fonts. It may be necessary to reinstall the server.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3709E HTCounter: Error loading font file, *font_name font_width* x *font_height*. Character *character_index*¨ *character_value* (0x:.*character_hex_value*X) is not printable.

Explanation: An error occurred while the htcounter program was attempting to load one of the fonts files. The font name, font size, and character in error are indicated in the message.

User Response: This error could occur if there was an installation problem, or if this file was inadvertently changed. Attempt to restore the original contents of the font files, Block1.dat and LCD.dat. It may be necessary to reinstall the server.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3710E HTCounter: ....expecting ServerRoot to be: *ServerRoot* (based on your server configuration file).

Explanation: An error occurred where the htcounter program expected ServerRoot in the httpd.conf configuration file to be set as indicated in the message. This message should be preceeded by another message with additional information.

User Response: Refer to the accompanying message to determine the cause of the problem.

IMW3711E HTCounter: No ServerRoot specified in httpd configuration file.

Explanation: The htcounter program relies on the setting of the ServerRoot directive in the httpd.conf configuration file to locate various files, such as the fonts files and the counter files. This message indicates that ServerRoot has not been set in the httpd configuration file.

User Response: Check the ServerRoot directive in the httpd.conf file. Set it to the current working directory of the Web server. The initial configuration file setting is /usr/lpp/internet/server_root.

IMW3712E HTCounter: there is an error with the HTCounter font files, see your Web system administrator.

Explanation: This message is displayed in the browser. An error was encountered by the htcounter program when trying to read the fonts files. There may be other messages in the server error log which may help determine the cause of the problem.

User Response: See your Web administrator. Check the httpd error log for the string `HTCounter:` and for other error messages that may indicate the cause of the problem.

IMW3713E HTCounter: Error reading counter file.

Explanation: A URL for this server specified the apicounter function available in the htcounter program. It tried to access a counter file that did not contain a count (for example, the counter file was not initialized properly).

User Response: Ask your Web administrator to use a text editor to initialize the counter file to the initial value you want to use (for example, 0).

IMW3714E HTCounter: Error flushing counter file.

Explanation: This is an internal error related to closing and flushing the counter file.

User Response: Use a text editor to view the counter file. Identify any problems accessing the file.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3715E HTCounter: Error closing counter file after reading.

Explanation: This is an internal error relating to closing and flusing the counter file.

User Response: Use a text editor to view the counter file. Identify any problem accessing the file.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

HTCounter Program messages

IMW3716E HTCounter: Error opening counter file for write.

Explanation: A URL for this server specified the apicounter function available in the htcounter program. It tried to access a counter file that did not have the permissions set properly. The httpd server needs write access to the file.

User Response: Ask your Web administrator to give the server write access to the file.

IMW3717E HTCounter: Error writing counter file.

Explanation: This is an internal error, relating to writing the counter file.

User Response: Use a text editor to view the counter file. Identify any problems accessing the file.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3718E HTCounter: Error closing counter file after writing.

Explanation: This is an internal error relating to closing and flusing the counter file.

User Response: Use a text editor to view the counter file. Identify any problems accessing the file.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3719E HTCounter: Counter file does not exist, tried: *counterfile*

Explanation: A URL for this server specified the apicounter function available in the htcounter program. It tried to access a non-existent counter file.

User Response: Ask your Web administrator to create the counter file in the counters subdirectory of the server's root directory with an initial value in it (typically, 0) and set permissions to allow the server write access.

IMW3720E HTCounter: Software error in accessing counter file.

Explanation: This is an internal error, relating to writing the counter file.

User Response: Use a text editor to view the counter file. Identify any problems accessing the file.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3721E HTCounter: No counter file specified on apicounter URL.

Explanation: You did not specify a counter file on your URL when trying to use the apicounter function.

User Response: Specify a valid counter file name on the URL.

IMW3722E HTCounter: No text specified on text2gif URL.

Explanation: You did not specify a text string on your URL when trying to use the text2gif function.

User Response: Specify a text string on the TEXT option on the URL.

IMW3723E HTCounter: ....errno indicates: *string*

Explanation: This message gives supplementary information for a problem indicated by the message that should have preceeded this one.

User Response: Refer to the accompanying message.

IMW3724E HTCounter: Not able to allocate memory for GIF tables.

Explanation: The htcounter program was not able to allocate storage for the temporary tables required to generate the gif image.

User Response: If the problem persists, there is a shortage of virtual storage. Attempt to use other utilities to determine if the server is getting enough storage. If the storage usage of the server contiually increases over time, there may be a problem with the server.

Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3725E HTCounter: Not enough memory to allocate Image buffer..we were trying for *number* **bytes.**

Explanation: The htcounter program was not able to allocate storage for the temporary tables required to generate the buffers required to generate the image.

User Response: If the problem persists, there is a shortage of virtual storage. Attempt to use other utilities to determine if the server is getting enough storage. If the storage usage of the server contiually increases over time, there may be a problem with the server.

Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3726E HTCounter: Not enough memory to allocate GIF buffer..we were trying for *number* bytes.

Explanation: The htcounter program was not able to allocate storage for the temporary buffers required to generate the gif image.

User Response: If the problem persists, there is a shortage of virtual storage. Attempt to use other utilities to determine if the server is getting enough storage. If the storage usage of the server contiually increases over time, there may be a problem with the server.

Contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3728E HTCounter: unable to open Fonts/FormsEtc.dat file. tried *path/filename*.

Explanation: The Web server HTCounter program could not open the file in this message.

User Response: Ensure that the file exists and that the HTCounter program has read access to the file. Update the file permissions, if necessary.

IMW3729E HTCounter: Not able to allocate memory for config_file in GetCounterFile().

Explanation: The Web server HTCounter program could not allocate enough memory to build *path/filename*.

User Response: The *path/filename* may be larger than the available resources or there is a shortage of virtual storage for the Web server.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW3730E HTCounter: ....errno= *return_code*, errno2= *reason_code*.

Explanation: An error occurred in the Web server HTCounter program.

The z/OS UNIX System Services *return_code* and *reason_code* may provide additional information about the cause of the problem. See "Explanation of errno and errno2 codes in messages" on page 605.

User Response: If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW4000E-4018E: HTIMAGE Messages

IMW4000E A URL was not returned, nor was the default set for the picture.

Explanation: The coordinates returned from the client by clicking on an image map are not defined in the corresponding map file. There is no default action defined in the map file. The htimage program could not determine a redirection URL to send back to the client.

User Response: Add a default to the map file for this picture.

IMW4001E An error occurred while parsing picture configuration file.

Explanation: The htimage program encountered unrecognized data in the map file.

User Response: Correct the map file.

IMW4002E Expecting a closing parenthesis.

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4003E Expecting a comma separating the x and y.

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4004E Expecting a coordinate pair.

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

HTIMAGE messages

IMW4005E Expecting a default URL.

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4006E Expecting a field name.

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4007E Expecting a first coordinate pair.

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4008E Expecting a radius.

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4009E Expecting a second coordinate pair.

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4010E Expecting a URL.

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4011E Expecting a y coordinate.

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4012E Expecting an x coordinate.

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4013E Picture configuration file was not found. Tried the following: *string.*

Explanation: The htimage program could not find the image map file referenced by the HTML page the client clicked on.

User Response: Correct the HTML or create the image map file.

IMW4014E Syntax error at line

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4015E The field name is not valid. Expecting *string*, **'default', 'rectangle', 'circle' or 'polygon'.**

Explanation: The htimage program found a syntax error in the map file.

User Response: Correct the map file.

IMW4016E The QUERY_STRING is not valid: *string.* **expecting either x,y or x=x&y=y;**

Explanation: The htimage program did not receive a valid coordinate pair for clicking on an image map. Verify that the query string was not damaged during the transmission of this request through your configuration file. Verify that the HTML page being viewed is correct. Verify that the browser program is operating correctly.

User Response: Correct any errors.

IMW4017E You did not set either the PATH_INFO or the PATH_TRANSLATED environment variable.

Explanation: The htimage program did not receive a map file name. Verify that the path info is not lost while translating your request through your configuration file. Verify that the HTML page being viewed is correct. Verify that the browser program is operating correctly.

User Response: Correct any errors.

IMW4018E You did not set the QUERY_STRING environment variable.

Explanation: The htimage program did not receive a QUERY_STRING. Verify that the query string is not lost while translating this request through your configuration file. Verify that the HTML page being viewed is correct. Verify that the browser program is operating correctly.

User Response: Correct any errors.

IMW5001E-5010E: HTADM Messages

IMW5001E Cannot open password file *string.*

Explanation: The htadm program was unable to open the specified password file. Verify that the file name is correct and that you have appropriate permissions.

User Response: Correct any problems found.

IMW5002I Administrative tool for Server access authorization.

Explanation: Usage information for htadm program.

User Response: None.

IMW5003E Cannot create file *name.*

Explanation: The htadm program could not create the password file. Verify the file system and permissions.

User Response: Correct any file system problems and retry.

IMW5004E Cannot create temporary file *name.*

Explanation: The htadm program could not create the password file. Verify the file system and permissions.

User Response: Correct any file system problems and retry.

IMW5005E Cannot find password file *string.*

Explanation: The htadm program could not find the referenced file. Verify the filename spelling and your permissions.

User Response: Correct any problems found and retry.

IMW5006I Correct

Explanation: The password that you checked is correct.

User Response: None.

IMW5007E Incorrect

Explanation: The password that you checked is not correct.

User Response: None.

IMW5008E Entry for user *name string* **in password file.**

Explanation: The htadm program found an error. For -adduser, the user name is already defined. For -deluser or -passwd, the user name is not found.

User Response: Correct the user name and retry.

IMW5009E File *name* **already exists.**

Explanation: The htadm program can't create the requested password file.

User Response: Correct the name and retry.

IMW5010E New file size *size* **does not match expected** *size.* **Original password file left intact.**

Explanation: The password file has been modified by someone else at the same time. Your change was not applied.

User Response: Try again.

IMW6102I-6805E: SSL Security Messages

IMW6102I Key File Password: *password*

Explanation: The key file password has not been entered.

User Response: Enter key file password.

IMW6104E Bad password

Explanation: The key file password you entered is not correct.

User Response: Verify that the password is correct and try entering it again. If the password cannot be remembered, a new key file has to be created.

IMW6304E Key Data Base not read. Please check key file existence, permission and ownership. Using Default root keys.

Explanation: An error was detected when the Web server attempted to open the key file specified in the configuration file.

User Response: Verify the key file has the correct read and write permissions. The Web server must have permission to read the key file.

IMW6310E SSL support initialization failed, server will run only in non-secure mode without listening on SSL port.

Explanation: A bad return code was found while checking one or more of the security components. Additional error messages may be issued to give you more information about the cause of the problem.

User Response: This error can occur if you specify a value on the SSLV2Timeout or SSLV3Timeout directive that is not within the valid range. If this is not the cause of the problem, a key, certificate, or password file may not be valid.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

Message IMW6802E may provide additional information to help you solve this problem.

IMW6801E Both SSL and normal modes have been turned off. At least one of the two must be turned on.

Explanation: An error occurred at server startup because both normal and SSL ports were disabled. At least one of the ports must be active to establish a TCP/IP socket connection.

User Response: For a secure network connection, set SSLMode on in the configuration file.

For more information, see "SSLMode - Turn SSL on or off" on page 550 and "NormalMode - Turn port on or off for HTTP connections" on page 546.

IMW6802E SSL Handshake failed: return code *return_code (description)*

Explanation: The SSL session could not be established due to an error during the SSL handshake. The GSK return code provides information on the reason for the failure.

User Response: Ensure that the Web server supports SSL and has a valid certificate. The GSK return code may provide additional information:

–1 GSK_ERROR_NO_CIPHERS
No ciphers were specified, or no common ciphers could be negotiated.

–2 GSK_ERROR_NO_CERTIFICATE
There is no client or server certificate. This error can occur if you have not defined a default key (certificate) for your key database. Check for a default key:

Use these steps for z/OS Version 1 Release 4 or later releases.
1. Start the gskkyman utility.
2. Select option 2 to open the key database.
3. Select option 9 to show the default key. If you do not have a default key listed, designate one of your keys as the default.

Use these steps for z/OS Version 1 Release 3 or earlier releases.
1. Start the gskkyman utility.
2. Select option 2 to open the key database.
3. Select option 7 to show the default key. If you do not have a default key listed, designate one of your keys as the default.

–4 GSK_ERROR_BAD_CERTIFICATE
The certificate is not valid.

–5 GSK_ERROR_INVALID_V2_HEADER
TCP/IP cannot process the SSL V2 header from the browser. The header is not valid because it does not comply with the architecture.

–6 GSK_ERROR_UNSUPPORTED_CERTIFICATE_TYPE
The certificate authority for the client or server certificate is not recognized.

–10 GSK_ERROR_IO
An I/O error occurred.

–11 GSK_ERROR_BAD_MESSAGE
Unrecognizable message from browser.

–14 GSK_ERROR_BAD_CERT_SIG
The signature on the certificate is not valid.

–15 GSK_ERROR_BAD_CERT
The client or server certificate is not valid.

–16 GSK_ERROR_BAD_PEER
The peer system is not recognized.

–17 GSK_ERROR_PERMISSION_DENIED
The transaction is not authorized.
–18 GSK_ERROR_SELF_SIGNED
The certificate was signed by the peer system.
–20 GSK_ERROR_BAD_MALLOC
A memory allocation failure occurred.
–21 GSK_ERROR_BAD_STATE
A bad state was detected for the session.
–22 GSK_ERROR_SOCKET_CLOSED
The secure session ended.
–23 GSK_ERROR_GSK_INITIALIZATION_FAILED
SSL initialization failed.
–24 GSK_ERROR_HANDLE_CREATION_FAILED
A socket data structure could not be created.
–25 GSK_ERROR_BAD_DATE
The time period of the certificate has expired.
–26 GSK_ERROR_BAD_KEY_LEN_FOR_EXPORT
Key length is not corrected for export version.
–27 GSK_ERROR_NO_PRIVATE_KEY
No private key is associated with this certificate.
–29 GSK_ERROR_INTERNAL
An internal error occurred.
–40 GSK_SOC_BAD_V2_CIPHER
V2 cipherspecs are incorrect.
–41 GSK_SOC_BAD_V3_CIPHER
V3 cipherspecs are incorrect.
–99 GSK_ERROR_UNKNOWN_ERROR
An unknown SSL error has occurred.
return_code **SSL error**
`SSL error` is used for an SSL return code other than the ones listed in this message description. For example, the SSL return code –12 `GSK_ERROR_BAD_MAC` is not included in the list above. If this error occurs, message IMW6802E will contain a return code of –12 and a description of `SSL error`.

For descriptions of GSK return codes, see the *z/OS System SSL Programming Guide and Reference*.

To access this book on the Web, go to the z/OS Book Server Web site at URL:

`http://www.ibm.com/servers/eserver /zseries/zos/bkserv/`

Note: The URL appears on two lines for printing purposes. Enter the URL in your browser as one line.

Return codes are listed with their descriptions in the /usr/lpp/gskss/include/gskssl.h directory.

If necessary, contact the IBM Software Support Center for assistance. For support information on the Web, see "Support services and resources" on page 604.

IMW6803I SSL Port

Explanation: Informational message displaying the SSL TCP/IP listen port.

User Response: None.

IMW6804E The parameter to -sslport option is not valid: *parameter*

Explanation: An invalid parameter was entered for sslport option on server startup. Default value is 443, but any port greater than 1024 may be specified.

User Response: Verify the parameter entered for sslport and try again.

IMW6805E The parameter to -sslmode option is not valid: *parameter*

Explanation: An invalid parameter was entered for sslmode option on server startup. Valid values are ON and OFF.

User Response: Verify the parameter entered for sslmode and try again.

Appendix G. TCP/IP reference

Selecting files or data sets 675
Data set search order 675
z/OS UNIX System Services data set environment. 675
TCP/IP file placement configuration 676

Selecting files or data sets

Data sets and files are comparable terms; ***file*** is the preferred term in z/OS UNIX. TCP/IP is a protocol that enables users to access data files or files.

If you are familiar with z/OS, you probably use the term “data set” to describe a unit of data storage. More specifically, a data set is the major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

If you are familiar with UNIX, you probably use the term “file” to describe a named set of records stored or processed as a unit.

Because TCP/IP is a protocol, it uses a set of semantic and syntactic rules to perform its communication functions and to access data sets and files. The TCP/IP rules use a search order to access the data sets or files. TCP/IP also uses data sets or files in a specific order to perform its communication functions that access other data sets and files. The TCP/IP search order for the data sets or files used or accessed by the OE applications differs from the search order for the data sets or files used or accessed by non-OE applications.

Some of the data sets or files have special importance because of their functions. For example, certain data sets are used for configuration.. The most fundamental data sets are the configuration data sets. In order, they are searched for the data set name of a service requested or a block of stored data. The search order differs between types of network configurations.

Data set search order

The table that follows illustrates the search order of data sets used by z/OS UNIX. Abbreviations used in this table are:

hlq Value of DATASETPREFIX from TCPIP.DATA. *hlq* defaults to TCPIP.

Note: z/OS UNIX does not support the TCP/IP post install ″zap″ to change this hlq.

$X Value of environment variable *X* used ″as is″ in fopen().

z/OS UNIX System Services data set environment

The following table maps TCP/IP specific MVS data sets to their counterparts in the z/OS UNIX environment.:

Table 10. z/OS UNIX System Services Environment

Data Set	z/OS UNIX
TCPIP.DATA	1. $RESOLVER_CONFIG (single user overrides) 2. /etc/resolv.conf (preferred OE default) 3. //SYSTCPD DD card 4. userID.TCPIP.DATA (single user overrides) 5. SYS1.TCPPARMS(TCPDATA) 6. TCPIP.TCPIP.DATA
STANDARD.TCPXLBIN	1. $X_XLATE 2. hlq.STANDARD.TCPXLBIN
HOSTS.SITEINFO	1. $X_SITE 2. /etc/hosts 3. HOSTS.SITEINFO 4. hlq.HOSTS.SITEINFO
HOSTS.ADDRINFO	1. $X_ADDR 2. /etc/hosts 3. HOSTS.ADDRINFO 4. hlq.HOSTS.ADDRINFO
ETC.PROTO	1. /etc/protocol 2. ETC.PROTO 3. hlq.ETC.PROTO
ETC.SERVICES	1. /etc/services 2. ETC.SERVICES 3. hlq.ETC.SERVICES

TCP/IP file placement configuration

The followingtable illustrates the recommended TCP/IP configuration file placement:

TCP/IP Configuration File	z/OS UNIX Applications
TCPIP.DATA share same file	SYS1.TCPPARMS(TCPDATA)
STANDARD.TCPXLBIN share same file	*hlq*.STANDARD.TCPXLBIN
HOSTS.SITEINFO share same file	HOSTS.SITEINFO *hlq*.HOSTS.SITEINFO
HOSTS.ADDRINFO share same file	HOSTS.ADDRINFO *hlq*.HOSTS.ADDRINFO
ETC.PROTO share same file	ETC.PROTO *hlq*.ETC.PROTO
ETC.SERVICES must not share same services file services are on different ports	/etc/services

Glossary

For more information on terms used in this book, go to the *IBM Dictionary of Computing* on the IBM Web site at URL:

http://www.ibm.com/networking/nsg/nsgmain.htm.

Bibliography

This bibliography lists the documentation related to Version 5.3 of the IBM HTTP Server for z/OS.

For a summary of available z/OS books and online information, see the *z/OS Information Roadmap*.

To access this documentation on the Web, go to the z/OS Book Server Web site at URL:

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

Note: The URL appears on two lines for printing purposes. Enter the URL on one line in your browser.

IBM HTTP Server

- *HTTP Server Planning, Installing, and Using*, SC34-4826-03

  This book explains how to plan for, install, configure, and use the Web server. The Programming section explains how to write external programs that interact with the Web server using the Common Gateway Interface (CGI), Go Webserver API (GWAPI), or LDAP API.
- *Troubleshooter*

  This Web-based guide provides the most current troubleshooting hints and tips for the Web server.
- *IBM Web Traffic Express for Multiplatforms User's Guide V1.0*, GC31-8645-00

  This book describes IBM's proxy server for the Web server.

  Note: The Web Traffic Express guide does not contain z/OS-specific information. Before using this guide, refer to the *HTTP Server Planning, Installing, and Using* book chapter on running your server as a proxy.

WebSphere Application Server

- *Application Server Planning, Installing, and Using*()
- *Troubleshooter*

  This Web-based guide provides troubleshooting hints and tips for the Application Server.

z/OS

- *z/OS Information Roadmap*, GC28-1727

  This book describes available information for the elements and features in z/OS. It also explains how to order z/OS documentation and how to access online z/OS information.
- *z/OS Planning for Installation*, GC28-1726

  This book lists the elements and features in z/OS. It explains how to get z/OS up and running, and provides information about migration actions for specific elements of z/OS.
- *z/OS MVS Planning: Workload Management*, GC28-1761

  This book explains Workload Management (WLM) concepts and interfaces, and includes the steps required for using WLM as well as its benefits.
- *z/OS UNIX System Services Planning*, SC28-1890

  This book explains how to plan for and install z/OS UNIX System Services.
- *z/OS UNIX System Services Command Reference*, SC28-1892

  This book explains the commands used by z/OS UNIX System Services.
- *z/OS MVS System Management Facilities*, GC28-1783
- *z/OS Cryptographic Services System SSL Programming Guide and Reference*, SC24-5877

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR

PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain translations, therefore, this statement may not apply to you.

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of the IBM HTTP Server.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:
- AIX
- BookManager
- CICS
- CICS/ESA
- C/MVS
- DFSMS
- IBM
- IBMLink
- Language Environment
- Multiprise
- MVS/ESA
- OpenEdition
- OS/2
- OS/390
- RACF
- RMF
- S/390
- System/390
- VTAM
- WebSphere
- z/OS

Adobe, Acrobat, and PostScript are trademarks of Adobe Systems Incorporated.

CORBA is a trademark of the Object Management Group, Incorporated.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Lotus, Domino, Lotus Go Webserver, and Lotus Notes are trademarks of the Lotus Development Corporation in the United States, or other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

Netscape and Netscape Navigator are trademarks of the Netscape Communications Corporation in the United States, or other countries, or both.

OpenView is a trademark of the Hewlett-Packard Company.

POSIX is a trademark of the Institute of Electrical and Electronic Engineers.

Tivoli, TME, TME10, TME 10 NetView, TME 10 Distributed Monitoring, are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

UNIX is a registered trademark of the Open Group in the United States and other countries.

Other company, product or service names may be the trademarks or service marks of others.

Index

Special Characters

.www_browsable files 463
%%CERTIF%% 22, 448
%%CLIENT%% 22, 448
%%SERVER%% 22, 448

Numerics

02AF (address space dirty) errors 24

A

ABEND, Web server
  CE dump requirements 7
  IMW0085E 612
  IMW0162E 617
  planning considerations 7
  Recovery directive 454
  recovery options 454
Access Control directives and subdirectives
  DefProt 436
  Protect 439
  Protection 443
  Protection subdirectives
    ACLOverride 445
    AuthType 445
    DeleteMask 445
    GetMask 445
    GroupFile 445
    Mask 446
    PasswdFile 446
    PostMask 447
    PutMask 447
    ServerID 447
    UserID 448
Access Control List (ACL) files 174
access control user IDs 22
access control using RACF or other SAF-based security products 8
access log
  maintenance options 192
  Server Activity Monitor 253, 303
  set filters 193
accessing the HTTP Server 43
AccessLog directive 492
AccessLogArchive directive 493
AccessLogExcludeMethod directive 494
AccessLogExcludeMimeType directive 495
AccessLogExcludeReturnCode directive 495
AccessLogExcludeURL directive 494
AccessLogExpire directive 495
AccessLogSizeLimit directive 496
AccessReportDescription directive 496
AccessReportDoDnsLookup directive 497
AccessReportExcludeHostName directive 498
AccessReportExcludeMethod directive 499
AccessReportExcludeReturnCode directive 500
AccessReportExcludeURL directive 497
AccessReportIncludeHostName directive 499
AccessReportIncludeURL directive 498
AccessReportRoot directive 500
AccessReportTemplate directive 501
AccessReportTopList directive 501
ACLOverride subdirective 445
activity statistics
  network 252
  server 250
AddBlankIcon directive 460
AddCharSet directive 523
AddClient directive 527
AddDirIcon directive 460
AddEncoding directive 523
AddIcon directive 461
AddLanguage directive 522
AddParentIcon directive 461
Address, Multiple IP 309
AddType directive 523
AddUnknownIcon directive 462
administration forms 50
AgentLog directive 493, 502
AlwaysWelcome directive 462
Anybody, Anyone, or Anonymous in Protection setups 164, 171, 172
ApplEnv directive 555
ApplEnvConfig directive 556
ApplEnvMax directive 558
ApplEnvMin directive 558
ApplEnvPrestart directive 559
application environments, WLM 228
  currently defined listing 229
  defining with ISPF panels 228
  ISPF panels 228
  modifying 231
  tasks to complete after defining 233
Application Server
  documentation 679
  migration considerations 15
  planning considerations 7
ASCII to EBCDIC 332, 334
ASCII/EBCDIC considerations, HTML pages 49
authentication 57, 363
Authentication directive 480
authorization 363
Authorization directive 479
authorizations for z/OS UNIX System Services 5
AuthType subdirective 445
automatic browser detection 527

B

backing up files 52
badredirect error code 472
badrequest error code 471
badscript error code 471
baduser error code 472
Basic directives
  BindSpecific 449
  Bounce 450
  DNS-Lookup 450
  HostName 451
  imbeds 451

Basic directives *(continued)*
NoLastMod 453
PidFile 453
Port 454
Recovery 454
ServerRoot 456
ServerToken 456
SysDumpName 456
UserId 457
BindSpecific directive 449
Bounce directive 450
browser requirements
Configuration and Administration Forms user interface 3
browsers, tested 3
byrule error code 472

C

Cache Accelerator 185
CacheAccessLog directive 502
CacheDefaultExpiry directory 529
CacheLastModifiedFactor directive 530
CacheLimit_2 directive 530
CacheLocalFile directive 566
CacheLocalMaxBytes directive 567
CacheLocalMaxFiles directive 567
CacheNoConnect directive 531
CacheOnly directive 531
CacheRoot directive 531
CacheSize directive 532
CacheUnused directive 532
Caching directive 533
central file management for PICS 289
certificate authority (CA)
acting as your own CA
buying CA software 63
HTTP Server CA utility
migration considerations 16
certificates
buying from a CA 62
digital 57
issuing your own 63
migrating 16
supported by the HTTP Server 62
CGI data manipulation 329
CGI example in C 335
CGI example in REXX 340
CGI parsing 332
CGI response 334
CGI scripts 315
CGI shell script example 342
CGI variables 331
cgi_error log 503
CgiErrorLog directive 503
cgiparse command 384
cgiutils 330
cgiutils command 387
Codepage type 366
codepage, language
DefaultFsCp 458
DefaultNetCp 459
command line, running web usage mining from 205
commands
cgiparse 384
cgiutils 387
htadm 389
htimage 221
commands *(continued)*
htlogrep 392
httpd 393
IMWHTTPD program 400
Simple Network Management Protocol (SNMP) 253
wwwcmd 409
community names, SNMP 276, 564
compiling CGI Programs 315
confidentiality, secure communications 56
Configuration and Administration Forms 50
configuration file
access control 435
basic server settings 449
directories and welcome page 459
error message customization 469
GWAPI application processing 477
logs and reports 491
meta-information 517
methods 518
performance settings 566
proxy server settings 529
resource mapping 536
security 545
syntax 294
timeouts 564
user directories 469
configuring a proxy server
caching proxy settings 303
port number 303
configuring the HTTP Server
controlling access to 51
editing the configuration file 50
using the Configuration and Administration Forms 50
CONNECT method 518
CONNECTION_POOL, httpd command 395
console commands
Workload Management 415
counter, how to display on a Web page 209
CounterDirectory directive 484
creating a
home page 49
creating PICS labels 291
criteria for rating Web sites 286
customizing your Web site
displaying page count, date, and time 209
sending customized pages 527
using server-side includes 214

D

DAEMON SAF facility class 5
data set
configuration file 9
DataFilter directive 482
date and time, how to display on a Web page 209
DBLookup directive 482
Debug Tool, for C/C++ GWAPI programs 375
DebugToolAddr directive 485
default
error messages 471
DefaultFsCp directive 458
DefaultNetCp directive 459
DefaultUser directive 318
DefineLBService paragraph, PICS 293, 295
DefineService paragraph, PICS 293, 294
DefProt directive 436
DELETE method 519

DeleteMask subdirective 445
digital certificate 57
digital signature 57
DirAccess directive 463
dirbrowse error code 472
directives
AccessLog 492
AccessLogArchive 493
AccessLogExcludeMethod 494
AccessLogExcludeMimeType 495
AccessLogExcludeReturnCode 495
AccessLogExcludeURL 494
AccessLogExpire 495
AccessLogSizeLimit 496
AccessReportDescription 496
AccessReportDoDnsLookup 497
AccessReportExcludeHostName 498
AccessReportExcludeMethod 499
AccessReportExcludeReturnCode 500
AccessReportExcludeURL 497
AccessReportIncludeHostName 499
AccessReportIncludeURL 498
AccessReportRoot 500
AccessReportTemplate 501
AccessReportTopList 501
AddBlankIcon 460
AddCharSet 523
AddClient 527
AddDirIcon 460
AddEncoding 523
AddIcon 461
AddLanguage 522
AddParentIcon 461
AddType 523
AddUnknownIcon 462
AgentLog 493, 502
AlwaysWelcome 462
ApplEnv 555
ApplEnvConfig 556
ApplenvMax 558
ApplenvMin 558
ApplEnvPrestart 559
Authentication 480
Authorization 479
BindSpecific 449
Bounce 450
CacheAccessLog 502
CacheDefaultExpiry 529
CacheExpiryCheck 530
CacheLastModifiedFactor 530
CacheLimit_2 530
CacheLocalFile 566
CacheLocalMaxBytes 567
CacheLocalMaxFiles 567
CacheNoConnect 531
CacheOnly 531
CacheRoot 531
CacheSize 532
CacheUnused 532
Caching 533
CgiErrorLog 503
CounterDirectory 484
DataFilter 482
DebugToolAddr 485
DefaultFsCp 458
DefaultNetCp 459
DefProt 436
directives *(continued)*
DirAccess 463
DirReadme 463
DirShowBrackets 464
DirShowBytes 464
DirShowCase 464
DirShowDate 464
DirShowDescription 465
DirShowGroup 465
DirShowHidden 465
DirShowIcons 465
DirShowMaxDescrLength 466
DirShowMaxLength 466
DirShowMinLength 466
DirShowMode 466
DirShowOwner 466
DirShowSize 467
Disable 519
DNS-Lookup 450
DoReporting 491
Enable 519
EnableFRCA 567
Enclave 568
Error 483
ErrorLog 503
ErrorLogArchive 504
ErrorLogExpire 505
ErrorLogSizeLimit 505
ErrorPage 470
Exec 537
ExecDirPass 539
Fail 539
FRCAAccessLog 569
FRCACacheEntries 570
FRCACacheOnly 570
FRCACacheSize 570
FRCAMaxFileSize 571
FRCANoCaching 571
FRCAStackName 571
FRCAVirtualHost 572
FRCAWLMParms 572
ftp_proxy 533
Gc 533
GcDailyGc 534
GcMemUsage 534
gopher_proxy 534
HostName 451
http_proxy 535
IconPath 467
imbeds 451
InheritEnv 544
InputTimeout 564
InstallPath 452
KeyFile 546
LDAPInfo 485
LiveLocalCache 573
Log 483
Log401Error 476
LogFormat 506
LoggingReportingDebugOutput 506
LoggingReportingProgram 509
LoggingReportingProgramOptions 509
LogTime 510
LogToSyslog 510
Map 540
MaxActiveThreads 574
MaxContentLengthBuffer 575

directives *(continued)*
MaxPersistRequest 576
MetaDir 517
MetaSuffix 518
NameTrans 478
no_proxy 535
NoCaching 535
NoLastMod 453
NoLog 511
NormalMode 546
ObjectType 480
OutputTimeout 565
overview 425
Pass 541
PersistTimeout 576
PICSDBLookup 482
PidFile 453
PluginDefault 560
PluginExclude 560
PluginInclude 561
Port 454
PostExit 484
PreExit 478
Protect 439
Protection 443
Protection subdirectives
ACLOverride 445
AuthType 445
DeleteMask 445
GetMask 445
GroupFile 445
Mask 446
PasswdFile 446
PostMask 447
PutMask 447
ServerID 447
UserID 448
ProxyAccessLog 303, 511
QOS 577
Recovery 454
Redirect 543
RefererLog 493, 512
ReportDataArchive 514
ReportDataCompressionProgram 512
ReportDataCompressionSuffix 513
ReportDataExpire 515
ReportDataSizeLimit 514
ReportDataUnCompressionProgram 513
ReportDataUnCompressionSuffix 513
ReportProcessOldLogs 513
ScriptTimeout 565
ServerInit 477
ServerPriority 577
ServerRoot 456
ServerTerm 484
ServerToken 456
Service 481
SMF 516
SMFRecordingInterval 516
SNMP 563
SNMPCommunityName 564
SocksServer 536
SSLCipherSpec 547
SSLClientAuth 549
SSLMode 550
SSLPort 550
SSLServerCert 551
directives *(continued)*
SSLV2Timeout 552
SSLV3Timeout 552
SSLX500CARoot 553
SSLX500Host 553
SSLX500Password 555
SSLX500Port 554
SSLX500UserID 554
SuffixCaseSense 529
SysDumpName 456
UseACLs 577
UseMetaFiles 578
UserDir 469
UserId 457
WebMasterEmail 564
Welcome 467
WLMClassify 478
WLMSN 578
Directory and Welcome Page directives
AddBlankIcon 460
AddDirIcon 460
AddIcon 461
AddParentIcon 461
AddUnknownIcon 462
AlwaysWelcome 462
DirAccess 463
DirReadme 463
DirShowBrackets 464
DirShowBytes 464
DirShowCase 464
DirShowDate 464
DirShowDescription 465
DirShowGroup 465
DirShowHidden 465
DirShowIcons 465
DirShowMaxDescrLength 466
DirShowMaxLength 466
DirShowMinLength 466
DirShowMode 466
DirShowOwner 466
DirShowSize 467
IconPath 467
UserDir 469
Welcome 467
directory listings 49
DirReadme directive 463
DirShowBrackets directive 464
DirShowBytes directive 464
DirShowCase directive 464
DirShowDate directive 464
DirShowDescription directive 465
DirShowGroup directive 465
DirShowHidden directive 465
DirShowIcons directive 465
DirShowMaxDescrLength directive 466
DirShowMaxLength directive 466
DirShowMinLength directive 466
DirShowMode directive 466
DirShowOwner directive 466
DirShowSize directive 467
Disable directive 519
Distinguished Name (DN) 57
DNS-Lookup directive 450
document root directory, path on Pass directive 543
domain name, specifying 451
DoReporting directive 491
dotdot error code 472

dynamic documents 316, 329

E

EBCDIC 332
EBCDIC conversion 332, 334
editing
  PICS configuration file 293
  server configuration file 50
Enable directive 519
EnableFRCA directive 567
Enclave directive 568
encryption
  hardware 65
  options 62
  overview 56
  SSL cipher specifications 64
  support 64
error condition key words
  badredirect 472
  badrequest 471
  badscript 471
  baduser 472
  byrule 472
  defined 470
  dirbrowse 472
  dotdot 472
  ipmask 472
  ipmaskproxy 473
  methoddisabled 473
  multifail 473
  noacl 473
  noentry 473
  noformat 476
  notallowed 473
  notauthorized 471
  notmember 471
  okredirect 471
  openfailed 473
  proxyfail 471
  scriptio 475
  scriptnotfound 475
  scriptstart 475
  service error 476
  setuperror 473
  unknownmethod 471
Error directive 483
error log
  maintenance options 195
  path 194
  specify path 194
error recovery, Web server
  CE dump requirements 7
  IMW0085E 612
  IMW0162E 617
  planning considerations 7
  Recovery directive 454
  recovery options 454
ErrorLog directive 503
ErrorLogArchive directive 504
ErrorLogExpire directive 505
ErrorLogSizeLimit directive 505
ErrorPage directive 470
example LDAP program 380
example, GWAPI REXX Service 373
examples
  accesses report (excluding beta and alpha7 requests) 202
examples *(continued)*
  changing the server's default encryption settings 160
  configuring log files 196
  department server accesses report (except internal addresses) 202
  page hits report 200
  protecting server resources 177
  PUT requests to beta subdirectory report 201
  setting up a secure server 69
  setting up SSL support for multiple IP addresses 159
Exec directive 537
ExecDirPass directive 539
execution modes, WLM
  defining application environments with ISPF 228
  establishing policies 228
  PROC 233
  running multiple servers 227
  scalable server subsystem 226
  standalone server 226
  using ISPF panels to define application environments. 228
export editions 64
External directive 319

F

Fail directive 539
Fast Response Cache Accelerator 185
FastCGI 317
  configuration changes 317
file permissions 466
filters for access log
  defaults 194
  external hits information 194
  overview 193
  reduce log size 193
  Website access information 194
forms, Configuration and Administration 50
FRCA, Fast Response Cache Accelerator 185
FRCAAccessLog directive 569
FRCACacheEntries directive 570
FRCACacheOnly directive 570
FRCACacheSize directive 570
FRCAMaxFileSize directive 571
FRCANoCaching directive 571
FRCAStackName directive 571
FRCAVirtualHost directive 572
FRCAWLMParms directive 572
Front Page
  accessing after installation 43
  viewing 43
ftp_proxy directive 533

G

Gc directive 533
GcDailyGc directive 534
GcMemUsage directive 534
GET method 519
GetMask subdirective 445
getting started 49
global settings for logs 191
gopher_proxy directive 534
group and user IDs
  access control user IDs
    %%CERTIF%% 22
    %%CLIENT%% 22

group and user IDs *(continued)*
access control user IDs *(continued)*
%%SERVER%% 22
IMWEB group ID 20
nonzero user ID 6, 20
surrogate user IDs, examples of
INTERNAL 22
PRIVATE 22
PUBLIC 22
WEBADM 22
WEBADM user ID 20
WEBSRV user ID 6, 20
group files, using 172
group-based statistics for web usage mining 204
GroupFile subdirective 445
guidelines for GWAPI programs 346
GWAPI
Debug Tool for C/C++ programs 375
environment variables 583
examples 346
programs 345
GWAPI configuration directives 360
GWAPI directives
Authentication 480
Authorization 479
CounterDirectory 484
DataFilter 482
DebugToolAddr 485
Error 483
Log 483
NameTrans 478
ObjectType 480
overview 477, 549
PICSDBLookup 482
plug-ins 560, 561
PostExit 484
PreExit 478
ServerInit 477
ServerTerm 484
Service 481
WLMClassify 478
GWAPI information 366
GWAPI process steps 348
Authentication 349
Authorization 349
Data Filter 349
Error 349
Log 349
Name Translation 349
Object Type 349
PICSDBLookup 349
PostExit 349
PreExit 349
Server Initialization 348
Server Termination 349
Service 349
WLMClassify 348
GWAPI REXX 366
accessing functions 367
errors 369
exec conditions 369
invoking 366, 367
link routines 371
specifying directives 368
GWAPI REXX exit conditions 373
GWAPI Service method handler 351

H

handling, signal 399, 408
hardware encryption 26, 65
HEAD method 519
headers
MIME headers 517
no-parse header programs 387
hints and tips
changing expired passwords 182
using the Server Activity Monitor 249
home (welcome) page 49
HostName directive 451
How do I?
configure the server 50
control access to administration 51
customize the Web server 20
develop servlets for the Web server 7
protect server resources 177
rate Web sites 286
serve directory listings 49
set up a secure server 69
start serving pages 49
use the Configuration and Administration Forms 293
how to rate a web site 291
htadm command 389
HTCodePage_t 366
HTCounter program 209
htimage command 221
htlogrep command 392
HTML documents
embedding CGI programs 326
HTML pages, ASCII/EBCDIC considerations 49
HTTP
disable methods 519
enable methods 519
headers 387
HTTP method handler 351
HTTP methods
CONNECT 518
DELETE 519
Disable 519
Enable 519
GET 519
HEAD 519
OPTIONS 519
POST 519
PUT 519
TRACE 519
HTTP return codes 353
HTTP Server 225
configuring
editing the configuration file 50
using the Configuration and Administration Forms 50
installing 19
managing 225
restarting, currently running server 405
security examples 69
stopping 45
viewing server's Front Page 43
http_proxy directive 535
httpd command 393
HTTPD_attributes() 354
HTTPD_authenticate() 354, 364, 365
HTTPD_exec() 356
HTTPD_extract() 355
HTTPD_file() 356
HTTPD_local_security() 358

HTTPD_log_access() 355
HTTPD_log_error() 357
HTTPD_log_trace() 357
HTTPD_proxy() 358
HTTPD_read() 357
HTTPD_restart() 358
HTTPD_reverse_translate() 355
HTTPD_set() 356
HTTPD_supply_label() 356
HTTPD_translate() 355
HTTPD_write() 357
httpd.conf configuration file 425

I

IconPath directive 467
IDINSEX tag 331
IDs
  access control user IDs
    %%CERTIF%% 22
    %%CLIENT%% 22
    %%SERVER%% 22
  IMWEB group ID 20
  nonzero user ID 6, 20
  surrogate user IDs, examples of
    INTERNAL 22
    PRIVATE 22
    PUBLIC 22
    WEBADM 22
  WEBADM user ID 20
  WEBSRV user ID 6, 20
Imbeds directive 451
IMWEB group ID 20
IMWHTTPD program 400
includes, server-side
  format 215
  preparing to use 214
  using 214
InheritEnv directive 544
InputTimeout directive 564
install path, Web server 20, 452
installing the HTTP Server
  installing for first time 20
  migrating 20
InstallPath directive 452
integrity, secure communications 56
IP Address, Multiple 309
IP address/host name protection 164
ipmask error code 472
ipmaskproxy error code 473

K

key database
  setting password 158
  storing password in a file 76, 79, 87, 100, 103, 110
KeyFile directive 546
keys
  asymmetric 56
  key pair 56
  key size 64
  private 56
  public 56
  symmetric 56

L

LABEL:CURRENT 380
LABEL:FIELD 380
LABELLIST 379
LabelsFor paragraph, PICS 293, 294
language and codepage
  DefaultFsCp 458
  DefaultNetCp 459
Languages for CGI programs 315, 332
LDAP
  configuring 59, 298
  overview 60
  search filters 297
  storing files on 300
  using with htadm command 389, 391
LDAP API 379
LDAPInclude directive 490
LDAPInfo directive 485
LDAPInfo subdirectives 486
  CacheTimeout 487
  ClientAuthType 488
  GroupNameFilter 490
  GroupSearchBase 490
  Host 486
  IdleConnTimeout 487
  KeyFileName 487
  Port 486
  SearchTimeout 487
  ServerAuthType 488
  ServerDN 488
  ServerPasswordStashFile 488
  Transport 486
  UserCertFilter 489
  UserNameFilter 489
  UserSearchBase 489
  WaitToRetryConnTime 487
LiveLocalCache directive 573
Local directive 318
Log directive 483
log maintenance options
  access log 192
  error log 195
log paths, specifying
  for error log 194
  for proxy log 192
Log401Error directive 476
LogFormat directive 506
logging and reporting
  changing default directory 35
  directives 491
  htlogrep command 392
  logs
    access log filters 193
    access log maintenance 192
    configuring scenario 196
    error log maintenance 195
    global settings 191
    overview 189
    path for error logs 194
    path for proxy logs 192
  reports
    overview 196
    sample scenarios 200
    update templates 199
    view default templates 200
    web usage mining statistics 203
  with system management facilities 206

Logging and Reporting directives
AccessLog 492
AccessLogArchive 493
AccessLogExcludeMethod 494
AccessLogExcludeMimeType 495
AccessLogExcludeReturnCode 495
AccessLogExcludeURL 494
AccessLogExpire 495
AccessLogSizeLimit 496
AccessReportDescription 496
AccessReportDoDnsLookup 497
AccessReportExcludeHostName 498
AccessReportExcludeMethod 499
AccessReportExcludeReturnCode 500
AccessReportExcludeURL 497
AccessReportIncludeHostName 499
AccessReportIncludeURL 498
AccessReportRoot 500
AccessReportTemplate 501
AccessReportTopList 501
AgentLog 493, 502
CacheAccessLog 502
CgiErrorLog 503
DoReporting 491
Errorlog 503
ErrorLogArchive 504
ErrorLogExpire 505
ErrorLogSizeLimit 505
FRCAAccessLog 569
LogFormat 506
LoggingReportingDebugOutput 506
LoggingReportingProgram 509
LoggingReportingProgramOptions 509
LogTime 510
LogToSyslog 510
NoLog 511
ProxyAccessLog 303, 511
RefererLog 493, 512
ReportDataArchive 514
ReportDataCompressionProgram 512
ReportDataCompressionSuffix 513
ReportDataExpire 515
ReportDataSizeLimit 514
ReportDataUnCompressionProgram 513
ReportDataUnCompressionSuffix 513
ReportProcessOldLogs 513
SMF 516
SMFRecordingInterval 516
LoggingReportingDebugOutput directive 506
LoggingReportingProgram directive 509
LoggingReportingProgramOptions directive 509
LogTime directive 510
LogToSyslog 510
LookAt online message facility 603
Lotus Notes adapter 358

M

management information base (MIB), SNMP 254
managing
PICS labels 287
managing PICS from central files 289
Map directive 540
mapping, resource
Exec 537
ExecDirPass 539
Fail 539
mapping, resource *(continued)*
InheritEnv 544
Map 540
Pass 541
Redirect 543
Mask subdirective 446
MaxActiveThreads directive 574
MaxContentLengthBuffer directive 575
MaxPersistRequest directive 576
messages, customizing
conditions, causes, and default messages 471
customization 469
ErrorPage directive 470
key word
badredirect 472
badrequest 471
badscript 471
baduser 472
byrule 472
defined 470
dirbrowse 472
dotdot 472
ipmask 472
ipmaskproxy 473
methoddisabled 473
multifail 473
noacl 473
noentry 473
noformat 476
notallowed 473
notauthorized 471
notmember 471
okredirect 471
openfailed 473
proxyfail 471
scriptio 475
scriptnotfound 475
scriptstart 475
service 476
setuperror 473
unknownmethod 471
Log401Error 476
overview 469
messages, looking up
Web server messages 603
z/OS LookAt online message facility 603
z/OS UNIX System Services errno and errno2 codes 605
Meta-information directives
MetaDir 517
MetaSuffix 518
MetaDir directive 517
MetaSuffix directive 518
method handler 351
methoddisabled error code 473
methods, HTTP
CONNECT 518
DELETE 519
Disable 519
Enable 519
GET 519
HEAD 519
OPTIONS 519
POST 519
PUT 519
TRACE 519
migrating your server 20
CA utility considerations 16

migrating your server *(continued)*
caching proxy server 15
proxy server 15
security changes 15
servlet support changes 15
MIME headers 329
multi-format processing 527
Multi-format Processing directives
AddCharSet 523
AddClient 527
AddEncoding 523
AddLanguage 522
AddType 523
SuffixCaseSense 529
multifail error code 473
Multiple IP Address 309
multiple servers and web usage mining 206
Multipurpose Internet Mail Extension (MIME) headers 517
MVS datasets, accessing 597

N

NameTrans directive 478
negative string 434
network activity statistics 252
no_proxy directive 535
no-parse header programs 387
noacl error code 473
NoCaching directive 535
noentry error code 473
noformat error code 476
NoLastMod directive 453
NoLog directive 511
NormalModel directive 546
North American edition 64
notallowed error code 473
notauthorized error code 471
notmember error code 471
nph headers 329
nph-programs 387, 389

O

object IDs, SNMP MIB 254
ObjectType directive 480
okredirect error code 471
openfailed error code 473
OPTIONS method 519
OutputTimeout directive 565
Overview of
LDAP 60
security concepts, options, and support 53
X.500 directory service 60
overview of CGI 315
overview of GWAPI 345

P

parsing CGI information on z/OS 332
Pass directive 541
PasswdFile subdirective 446
password
authorized 50
changing expired 182
community name, SNMP 276, 564
files 389
password *(continued)*
key database
specifying 158
storing in a file 76, 79, 87, 100, 103, 110
path-based statistics for web usage mining 204
performance
recommendations on method for starting the Web server 39
suggestions if using RACF or another SAF-based security product 8
PersistTimeout directive 576
PICS 482
PICS for rating services and label bureaus 288
PICS for web site administrators 287
PidFile directive 453
planning for installation 3
Platform for Internet Content Selection (PICS) 285
central files 289
creating labels 291
DefineLBService paragraph 293, 295
DefineService paragraph 293, 294
definition 285
editing PICS configuration files 293
filtering content 303
LabelsFor paragraph 293, 294
PICS label retrieval directive 482
rating files 291
starting a PICS service 290
storing on server 290
syntax 294
updating configuration file 293
using online Configuration and Administration Forms 293
using wildcards 295
PluginDefault directive 560
PluginExclude directive 560
PluginInclude directive 561
Port directive 454
port number for proxy server 303
porting CGI pograms 362
positive string 434
POST method 519
POST, HTTP method 519
PostExit directive 484
PostMask subdirective 447
predefined functions 354
PreExit directive 478
PROC, WLM 409
process steps, HTTP Server 477
Protect directive 439
protecting server resources 177
protecting your CGI programs 331
protection
default protection with %%CLIENT%% 176
options 163
Protection directive 443
Protection subdirectives
ACLOverride 445
AuthType 445
DeleteMask 445
GetMask 445
GroupFile 445
Mask 446
PasswdFile 446
PostMask 447
PutMask 447
ServerID 447
UserID 448

proxy
  ftp_proxy 303, 533
  gopher_proxy 303, 534
  http_proxy 303, 535
  no_proxy 535
  proxy server 303
proxy log path 192
proxy server
  configuring
    designating port number 303
    specifying protocols 303
  overview 303
Proxy Server settings directives
  CacheDefaultExpiry 529
  CacheExpiryCheck 530
  CacheLastModified Factor 530
  CacheLimit2 530
  CacheNoConnect 531
  CacheOnly 531
  CacheRoot 531
  CacheSize 532
  CacheUnUsed 532
  Caching 533
  ftp_proxy 533
  Gc 533
  GcDailyGc 534
  GcMemUsage 534
  gopher_proxy 534
  http_proxy 535
  MaxContentLengthBuffer 575
  no_proxy 535
  NoCaching 535
  ProxyAccessLog 303, 511
  SocksServer 536
ProxyAccessLog directive 303, 511
proxyfail error code 471
PUT
  method 519
  with server 519
PUT, HTTP method 519
PutMask subdirective 447

Q

QOS directive 577
QUERY_STRING 328

R

RACF 8
  02AF (address space dirty) errors 24
  defining
    IMWEB group ID 20
    SURROGAT class support for surrogate IDs 23
    WEBADM user ID 20
    WEBSRV user ID 20, 29
  Lotus Notes Adapter function permission 28
  performance suggestions 8
  planning considerations 8
  program control, turning on 24
  SMF permission 28
  WEBSRV permission 23
  Workload Management permission 27
rating
  criteria for Web sites 286
  Web servers 285
rating file for PICS 289
record formats, SMF
  configuration 279
  header 279
  performance 281
Recovery directive 454
Redirect directive 543
referencing CGI programs in HTML 326
RefererLog directive 493, 512
remote configuration and administration 50
report filters
  external hits information 198
  most visted Web pages 199
  reduce report scope 198
  Website access information 198
ReportDataArchive directive 514
ReportDataCompressionProgram directive 512
ReportDataCompressionSuffix directive 513
ReportDataExpire directive 515
ReportDataSizeLimit directive 514
ReportDataUnCompressionProgram directive 513
ReportDataUnCompressionSuffix directive 513
reporting and logging
  changing default directory 35
  directives 491
  htlogrep command 392
  logs
    access log filters 193
    access log maintenance 192
    configuring scenario 196
    error log maintenance 195
    global settings 191
    overview 189
    path for error logs 194
    path for proxy logs 192
  reports
    overview 196
    sample scenarios 200
    update templates 199
    view default templates 200
    web usage mining statistics 203
  with system management facilities 206
reporting program
  DoReporting directive 491
  examples 200
  HTLOGREP 197, 509
  LoggingReportingProgram directive 509
  LoggingReportingProgramOptions directive 509
  templates 199
  third-party, using 196, 509
  Web Usage Mining 197
ReportProcessOldLogs directive 513
reports
  overview 196
  sample scenarios 200
  update templates 199
  view default templates 200
  web usage mining statistics 203
resource mapping
  Exec 537
  ExecDirPass 539
  Fail 539
  InheritEnv 544
  Map 540
  Pass 541
  Redirect 543
response generation on z/OS 334

restarting
  currently running server 405
return codes from predefined functions 359
returning CGI output 331
running webusage mining from the command line 205

S

scriptio error code 475
scriptnotfound error code 475
scriptstart error code 475
ScriptTimeout directive 565
secure communications, concepts 55
Secure Sockets Layer (SSL)
  client authentication 58
  encryption levels 64
  overview 58
  support for multiple IP addresses 62
  tunneling 303
security
  certificate authorities supported 62
  cipher specifications supported 64
  concepts
    authentication 55
    certificate revocation list (CRL) 58
    cipher specifications 64
    confidentiality 56
    digital certificate 57
    digital signature or ID 57
    Distinguished Name 57
    encryption 56
    HTTPS protocol 58
    integrity 56
    Lightweight Directory Access Protocol (LDAP) 58
    overview 55
    Public Key Infrastructure (PKI) 58
    secure communication, characteristics 55
  configuring
    encryption settings for the server 160
    protection for server resources 177
    secure connections 69
    SSL support for multiple IP addresses 159
  encryption support 64
  error
    asn.1 158
  examples 69
  hardware encryption support 26, 65
  options
    overview 61
    protecting server resources 163
    sending different certificates from each server IP addresses 62
    setting up secure connections 61
    specifying the encryption level to be used 62
    using S/390 cryptographic hardware 65
Security directives
  KeyFile 546
  NormalMode 546
  SSLCipherSpec 547
  SSLClientAuth 549
  SSLMode 550
  SSLPort 550
  SSLServerCert 551
  SSLV2Timeout 552
  SSLV3Timeout 552
  SSLX500CARoot 553
  SSLX500Host 553
Security directives *(continued)*
  SSLX500Password 555
  SSLX500Port 554
  SSLX500UserID 554
self-signed certificates, example 84, 107, 130
sending information 328
server
  installing 19
  stopping 45
  viewing server's Front Page 43
Server Activity Monitor
  access log 253
  network activity statistics 252
  server activity statistics 250
  using 249
server request process 347
SERVER SAF facility class 5
server-side includes
  format 215
  preparing to use 214
  using 214
ServerID subdirective 447
ServerInit directive 477
ServerPriority directive 577
ServerTerm directive 484
ServerToken directive 456
Service directive 481
service unavailable 476
serving directory listings 49
serving pages 49
servlet support 15
setup.sh command
  common error messages 32
  format and options 30
setuperror error code 473
setuptgt.sh command
  common error messages 34
  format and options 33
signal handling 399, 408
Simple Network Management Protocol (SNMP) 253
  commands and protocol 253
  community names 276, 564
  e-mail address for problem reports 276, 564
  Examples for single Web server 270
  Mangement Informatin base(MIB) file sample 269
  Multiple Web servers 273
  object IDs and variable names 254
  object IDs for management information base (MIB) 254
  overview 253
  password 276, 564
  problem reports, email address 276, 564
  receiving problem reports 276, 564
  turning support on and off 277, 563
SMF directive 516
SMFRecordingInterval directive 516
SNMP directive 563
SNMPCommunityName 564
socksifying a proxy server 303
SocksServer directive 536
specifying log paths
  for error log 194
  for proxy log 192
SSL 58
SSLCipherSpec 547
SSLClientAuth directive 549
SSLMode directive 550
SSLPort directive 550

SSLServerCert 551
SSLV2Timeout 552
SSLV3Timeout 552
SSLX500CARoot 553
SSLX500Host 553
SSLX500Password 555
SSLX500Port 554
SSLX500UserID 554
starting a PICS service 290
starting and stopping your server
  starting at initialization 39
  starting automatically 39
  starting from httpd command 39
  starting from OE shell 39
  starting multiple instances 40
  stopping 45
static documents 316, 329
storing files on server 290
string comparison 333
SuffixCaseSense directive 529
superuser 5
SUPERUSER SAF facility class 5
surrogate user IDs 22
SysDumpName directive 456
System Management directives
  ApplEnv 555
  ApplEnvConfig 556
  ApplEnvMax 558
  ApplEnvMin 558
  ApplEnvPrestart 559
  SNMP 563
  SNMPCommunityName 564
  WebMasterEmail 564
System Management Facilities (SMF)
  controlling logging interval 278
  logging information with 206
  overview 277
  record formats
    configuration 279
    header 279
    performance 281
  turning on and off 278
    with the console MODIFY command 278

T

TCP/IP
  autostarting the Web server with 29
  customizing 28
  verifying name resolution 29
tested browsers 3
threads
  hints and tips 574
  MaxActiveThreads directive 574
  server use of 566
time specifiers 435
Timeout directives
  InputTimeout 564
  OutputTimeout 565
  PersistTimeout 576
  ScriptTimeout 565
TRACE method 519
transience 370
troubleshooting
  error recovery options 454
Tuning directives
  CacheLocalFile 566
Tuning directives *(continued)*
  CacheLocalMaxBytes 567
  CacheLocalMaxFiles 567
  Enclave 568
  Fast Response Cache Accelerator (FRCA) directives
    EnableFRCA 567
    FRCAAccessLog 569
    FRCACacheEntries 570
    FRCACacheOnly 570
    FRCACacheSize 570
    FRCAMaxFileSize 571
    FRCANoCaching 571
    FRCAStackName 571
    FRCAVirtualHost 572
    FRCAWLMParms 572
  LiveLocalCache 573
  MaxActiveThreads 574
  MaxPersistRequest 576
  PersistTimeout 576
  QOS 577
  ServerPriority 577
  UseACLs 577
  UseMetaFiles 578
  WLMSN 578

U

unknownmethod error code 471
updating PICS configuration file 293
UseACLs directive 577
UseMetaFiles directive 578
user and group IDs
  access control user IDs
    %%CERTIF%% 22
    %%CLIENT%% 22
    %%SERVER%% 22
  IMWEB group ID 20
  nonzero user ID 6, 20
  surrogate user IDs, examples of
    INTERNAL 22
    PRIVATE 22
    PUBLIC 22
    WEBADM 22
  WEBADM user ID 20
  WEBSRV user ID 6, 20
user name and password protection 163
user-based statistics for web usage mining 204
UserDir directive 469
UserID directive 457
UserID subdirective 448
uses for CGI 316
usr/lpp/internet, default Web server install path 20, 452
usr/lpp/internet/server_root, default Web server working directory 456

V

variable names, SNMP MIB 254
virtual hosts 309

W

Web site rating criteria 286
Web Traffic Express 303
web usage mining report
  group-based statistics 204

web usage mining report *(continued)*
multiple servers 206
overview 203
path-based statistics 204
running from the command line 205
user-based statistics 204
WEBADM user ID 20, 35
WebMasterEmail directive 564
WebSphere Application Server
documentation 679
migration considerations 15
planning considerations 7
WEBSRV user ID 6, 20
welcome (home) page 49
Welcome directive 467
welcome page and directory directives
AddBlankIcon 460
AddDirIcon 460
AddIcon 461
AddParentIcon 461
AddUnknownIcon 462
AlwaysWelcome 462
DirAccess 463
DirReadme 463
DirShowBrackets 464
DirShowBytes 464
DirShowCase 464
DirShowDate 464
DirShowDescription 465
DirShowGroup 465
DirShowHidden 465
DirShowIcons 465
DirShowMaxDescrLength 466
DirShowMaxLength 466
DirShowMinLength 466
DirShowMode 466
DirShowOwner 466
DirShowSize 467
IconPath 467
UserDir 469
Welcome 467
WLMClassify directive 478
WLMSN directive 578
Workload management (WLM)
application environments, defining 231
classification rules, defining 231
completing the configuration process 233
console commands 415
defining application environments 228
directives
ApplEnv 555
ApplEnvConfig 556
FRCAWLMParms 572
WLMClassify 478
enabling Web server support 27
example: multiple scalable servers on a system 241
example: single scalable server on a system 235
execution modes
running multiple servers 227
Scalable Server mode 226
Standalone mode 226
overview 228
policies, establishing 228
service definition, choosing 228
tasks after defining application environments and classification rules 233
using ISPF panels 228
writing GWAPI programs 346
writing GWAPI REXX 369
wwwcmd 409

X

X.500 directory service 60

Z

z/OS Debug Tool, for C/C++ GWAPI programs 375
z/OS LookAt online message facility 603
z/OS UNIX System Services
authorizations for 5
superusers 5

Program Number: 5694-A01

Printed in U.S.A.

SC34-4826-03