

IBM HTTP Server on z/OS Migrating from Domino-powered to Apache-powered

Edward McCarthy



z Systems



International Technical Support Organization

**IBM HTTP Server on z/OS: Migrating from
Domino-powered to Apache-powered**

October 2016

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

Third Edition (October 2016)

This edition applies to Version 9 of IBM HTTP Server powered by Apache (product number xxx-xxx).

This document was created or updated on July 12, 2017.

© Copyright International Business Machines Corporation 2013, 2016. All rights reserved.

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

Contents

Notices	ix
Trademarks	x
IBM Redbooks promotions	xi
Preface	xiii
Authors	xiii
Now you can become a published author, too!	xiv
Comments welcome	xiv
Stay connected to IBM Redbooks	xiv
Summary of changes	xv
October 2016, Third Edition	xv
Chapter 1. Introduction to IBM HTTP Server for z/OS	1
1.1 Why you should migrate	2
1.1.1 HTTP Server terminology	3
1.1.2 IBM statement of support	3
1.1.3 Documentation	3
1.2 New features in V8.5.5	4
1.2.1 Updates to this IBM Redpaper publication	4
1.3 New features in V9.0.0	5
1.4 Determining which IBM HTTP Server you are running	6
1.4.1 Using SDSF to find running HTTP Servers	6
1.4.2 Determining which TCP/IP ports are used	7
1.4.3 Accessing the home page	8
1.4.4 Using the ps command to check for HTTP Servers	8
1.4.5 Checking for non-running IBM HTTP Servers	9
1.5 Checking your IBM HTTP Server version	10
1.5.1 Determining IBM HTTP Server powered by Domino version	10
1.5.2 Determining IBM HTTP Server powered by Apache version	12
1.5.3 For more information	12
Chapter 2. Features and performance	13
2.1 New features in V8.5.5	14
2.1.1 Listing MVS data sets	14
2.1.2 HTTP response translation improvements	15
2.1.3 Federal Information Processing Standards (FIPS140-2) support	16
2.1.4 31-bit support	17
2.1.5 Features in IHS powered by Domino and not in IHS powered by Apache	17
2.2 Support for zEnterprise Data Compression	17
2.2.1 zEnterprise Data Compression requirements	18
2.2.2 Verifying that zEnterprise Data Compression is active	18
2.2.3 Enabling use of zEnterprise Data Compression	19
2.2.4 Testing	19
2.2.5 SMF information	20
2.2.6 Comparing results	20
2.2.7 SMF information about hardware compression	21
2.2.8 Compression log usage information	23

2.3	Functional differences	23
2.4	Performance comparison	24
2.4.1	Basic measure of throughput test	24
2.4.2	CPU utilization test	25
2.4.3	Measure of throughput with and without caching test	26
2.4.4	Measure of CPU test	27
Chapter 3.	Installing your first IHS	29
3.1	IHS code that is shipped with z/OS 2.2.	30
3.2	Obtaining and installing the product code	30
3.2.1	Delivered as a component of other IBM products	30
3.2.2	Downloaded at no charge from the IBM Shopz website	30
3.3	Ordering and installing by using Shopz.	31
3.3.1	IBM Shop z website	31
3.3.2	Ordering software	31
3.3.3	Downloading the software.	38
3.3.4	FTP product code to z/OS UNIX in z/OS	40
3.3.5	First job to run: GIMUNZIP	44
3.3.6	Second job to run: UNZIPJCL.	45
3.3.7	Setting up SMP/E	46
3.3.8	Receiving the product code	50
3.3.9	Applying the product code	50
3.3.10	Accepting the product code	52
3.3.11	Summary.	52
3.4	Installation when a component of another IBM product	52
3.5	Sample real-world setup process	53
3.5.1	Defining a configuration directory	53
3.5.2	Defining a user ID	54
3.5.3	Defining a protected user ID for the started task.	55
3.5.4	Creating the IHS	56
3.5.5	Defining a RACF STARTED rule	56
3.5.6	Creating a Started Task to run the IHS.	57
3.5.7	Verifying that IHS is working.	57
3.6	Using intermediate symbolic links	58
3.6.1	Setting up an intermediate link	59
3.7	Maintenance upgrade	60
3.7.1	Gradual maintenance rollout approach.	61
3.7.2	New_install_root shell	62
Chapter 4.	Administration	65
4.1	Running IBM HTTP Server powered by Apache.	66
4.2	Using started tasks	66
4.2.1	Starting the server.	66
4.2.2	Stopping the server.	67
4.2.3	Recycling the server to pick up changes.	67
4.2.4	Modifying command support in V8.5.5	68
4.2.5	Displaying version in job log	70
4.3	Using apachectl from the command line	71
4.3.1	Starting the server.	71
4.3.2	Stopping the server.	72
4.3.3	Restarting the server.	72
4.3.4	Mix and match.	72
4.4	Integration with WebSphere Application Server	72

4.5 Configuration	73
4.5.1 Listen directive	73
4.5.2 Virtual hosting	73
4.6 Monitoring	75
4.6.1 SDSF	75
4.6.2 Checking pid and log files	75
4.6.3 Server status	76
4.6.4 Server status by using the modify command	77
4.6.5 Thread usage	77
4.7 Diagnostic tools and information	78
4.8 Troubleshooting	79
4.9 Migrating previous versions	79
4.10 Tracing	80
4.10.1 Information about tracing	80
4.10.2 Limitation	81
4.10.3 Some examples	81
4.11 Handling logging	82
4.12 Macro support	82
4.13 Conditional controls	83
Chapter 5. Migration	85
5.1 Planning your migration	86
5.1.1 Migration plan	86
5.2 Migration guidance	87
5.2.1 Scalable mode	87
5.2.2 SMF records	87
5.2.3 Server home directory	88
5.2.4 Ports	88
5.2.5 Virtual hosts	89
5.2.6 Security	90
5.2.7 Logging	91
5.2.8 URL and file mapping directives	92
5.2.9 WebSphere Application Server plug-in	94
5.2.10 Timeouts	94
5.2.11 Caching	95
5.2.12 ASCII/EBCDIC considerations	95
5.2.13 GWAPI	97
5.2.14 Reverse Proxy	97
5.2.15 Comparing DGW and IHS use of directives	97
5.2.16 Cleaning up PARMLIB	97
5.3 Migrating Library Server	97
5.3.1 Set up in DGW	98
5.3.2 Set up in V8.5.5	98
5.3.3 Testing Library Server	101
Chapter 6. Scalability and workload management.	103
6.1 Overview	104
6.2 DGW approach	104
6.3 IHS V8.5.5 approach	105
6.3.1 Multi-processing module	105
6.3.2 How V8.5.5 looks on z/OS	107
6.3.3 Example of dynamic scalability	108
6.3.4 Sizing your server	109

6.4	V8.5.5 support for WLM	110
6.5	Working with WLM in IHS V8.5.5	111
6.5.1	Mapping app requests to one WLM transaction class as default approach	111
6.5.2	Mapping an application for a specific virtual host	111
6.5.3	Mapping multiple applications within a specific virtual host	111
6.5.4	Connecting WLM directives and WLM setup	112
6.5.5	WLM in action	113
6.6	Summary	116
Chapter 7	Security	117
7.1	Security overview	118
7.2	Configuring V8.5.5 for your security requirements	118
7.2.1	Allowing unauthenticated access	119
7.2.2	Allowing all authenticated user access	119
7.2.3	Allowing authenticated user that belongs to a group access	120
7.2.4	Allowing authenticated user access with client credentials	121
7.2.5	Required SAF definitions	121
7.2.6	Complex authorization logic	122
7.3	SSL and Session ID	122
7.4	Configuring SSL support	123
7.4.1	RACF or keystore files	123
7.4.2	Creating required certificates	123
7.4.3	Updating httpd.conf	124
7.4.4	Testing SSL	125
7.4.5	Advanced SSL options	126
7.4.6	Basic SNI Support	126
7.5	Controlling access by using mod_rewrite	128
7.6	Caching and security considerations	129
7.6.1	Authorization and access control	129
7.6.2	Local vulnerabilities	130
7.6.3	Cache poisoning	130
Chapter 8	System Management Facilities support in IHS V8.5.5	131
8.1	SMF overview	132
8.2	DGW and SMF	132
8.3	V8.5.5 and SMF	132
8.3.1	Comparing DGW and V8.5.5 SMF records	132
8.3.2	Content	133
8.3.3	SMF browser	134
8.3.4	Enabling for subtype 13	134
8.3.5	Enabling for subtype 14	135
8.4	Summary	136
Chapter 9	Plug-in for WebSphere Application Server	137
9.1	Plug-in overview	138
9.2	Intelligent Management for Web Servers feature	138
9.3	Configuring WebSphere Application Server plug-in into IBM HTTP Servers	140
9.3.1	IBM HTTP Server powered by Domino	140
9.3.2	IBM HTTP Server powered by Apache	140
9.3.3	Key difference	141
9.3.4	Working with the plug-in configuration file	141
9.3.5	Regenerating the plug-in configuration file	143
9.3.6	Managing who serves application static files	143

Chapter 10. Cache configuration	145
10.1 Caching overview	146
10.1.1 What can be cached	146
10.1.2 Not cached	147
10.1.3 File-handle caching	147
10.1.4 In-memory caching	148
10.1.5 Disk-based caching	150
10.2 Fast Response Cache Accelerator	151
Chapter 11. Modules	153
11.1 Why custom modules are used	154
11.1.1 Popularity of Apache modules	154
11.2 DGW modules	154
11.2.1 Migrating GWAPI modules to V8.5.5 modules	154
11.3 Simple helloworld module	155
11.3.1 Code structure of helloworld module	155
11.3.2 Compiling the helloworld module	156
11.3.3 Integrating the new helloworld module into the configuration file	158
11.3.4 Testing the helloworld module	158
11.4 Apache-supplied example module	158
11.4.1 Code structure overview	158
11.4.2 Compiling the example module	159
11.4.3 Integrating the example_module into the server conf file	159
11.4.4 Testing the example_module	160
11.5 Using an open source Apache module	160
11.5.1 Limit IP module	161
11.5.2 Compiling the module	161
11.5.3 Updating the httpd.conf file	161
11.5.4 Restarting and testing	161
Chapter 12. CGI scripts	163
12.1 CGI overview	164
12.1.1 Brief history	164
12.1.2 CGI disadvantage	164
12.1.3 CGI alternatives	164
12.1.4 A use for CGI	164
12.2 Rexx CGI programs in DGW	165
12.2.1 DGW support for CGI programs	165
12.2.2 Sample Rexx CGI program	165
12.2.3 Using exec directive	165
12.2.4 Running the example.rx CGI	165
12.3 Rexx CGI programs in V8.5.5	166
12.3.1 Default cgi-bin setup	166
12.3.2 Changing example.rx to enable it for V8.5.5	166
12.3.3 Support for cgiutils and cgiparse in V8.5.5.2	169
12.3.4 Escaped characters	170
12.3.5 Rexx CGI summary	172
12.3.6 More complex Rexx sample	172
12.4 Perl CGI programs in V8.5.5	173
12.4.1 Using Perl on z/OS	173
12.4.2 Sample Perl CGI program	173
12.4.3 IHS and LIBPATH	173
12.4.4 Testing the Perl CGI program	174
12.5 PHP CGI programs in V9	174

12.5.1	Using php on z/OS	174
12.5.2	Rocket PHP software	174
12.5.3	Running PHP CGI programs	174
12.5.4	PHP by using the action approach	175
12.5.5	PHP by using the shebang approach	181
12.6	PHP CGI programs in V8.5.5	185
12.6.1	Sample php CGI program	185
12.6.2	PHP wrapper program	185
12.6.3	Modifications to the httpd.conf file	186
12.6.4	Testing the PHP CGI program	186
12.7	Lua support	186
12.7.1	Lua overview	186
12.7.2	Lua and Apache server	186
12.7.3	Lua advantage	187
12.7.4	Using Lua	187
12.7.5	Lua examples	187
12.7.6	More information	188
Appendix A. Additional material		189
Locating the web material		189
Using the web material		189
System requirements for downloading the web material		189
Downloading and extracting the web material		190

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

AIX®	MVS™	RMF™
CICS®	RACF®	System z®
Domino®	Redbooks®	WebSphere®
IBM®	Redpaper™	z/OS®
Lotus®	Redbooks (logo)  ®	zEnterprise®

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Find and read thousands of IBM Redbooks publications

- ▶ Search, bookmark, save and organize favorites
- ▶ Get personalized notifications of new content
- ▶ Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App



Download
Now

Android



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



ibm.com/Redbooks

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

Users of IBM® z/OS® for the past several years had a choice of two HTTP Servers that they can use. Now, one server became strategic while the other is no longer supported with z/OS V2.2. IHS powered by Apache supports IPv6 and 64-bit execution and includes security authentication and authorization capabilities that are similar to those capabilities that are provided in IHS powered by IBM Domino®.

This IBM Redpaper™ publication is aimed at technicians who are responsible for planning and deploying system software. It provides information on about the various features that are available in IBM HTTP Server powered by Apache. It also provides guidance about how to upgrade from the old product to the new product.

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Edward McCarthy is an IBM certified specialist with over 14 years experience working with IBM WebSphere® Application Server on various operating systems, including z/OS, Linux on IBM System z®, IBM AIX®, and Windows. He has designed and configured numerous WebSphere Application Server environments for many customers. He also has been involved with all aspects of WebSphere Application Server, such as tuning, automating administration, problem solving, and integration. Before joining IBM in 2000, he was an IBM CICS® and WebSphere MQ systems programmer with an Australian government department for over nine years. During this time, he implemented each new CICS version as part of an IBM beta program. Edward also has worked on several IBM Redbooks®. He has presented WebSphere topics at various conferences.

Thanks to the following people for their contributions to this project:

- ▶ Rich Conway
- ▶ Gary Puchkoff
- ▶ Donald Calas
- ▶ Mike Cox
- ▶ Eric M Covener
- ▶ Peter Kingsley
- ▶ Patrick O'Donnell
- ▶ Jeff Mierzejewski
- ▶ Bob Rinda
- ▶ Marna Walle
- ▶ William White
- ▶ Keith Winnard

Thanks to the authors of the first edition, *IBM HTTP Server on z/OS: Migrating from Domino-powered to Apache-powered*, published in October 2013:

- ▶ Mike Ebberts
- ▶ Douglas Cardoso
- ▶ Camila Colanica
- ▶ Edward McCarthy
- ▶ Calalin Mierlea

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>

Summary of changes

This section describes the technical changes that were made in this edition of the paper and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes

for *IBM HTTP Server on z/OS: Migrating from Domino-powered to Apache-powered*
as created or updated on July 12, 2017.

October 2016, Third Edition

This revision includes the following new and changed information.

New information

This revision describes new capability introduced in IBM HTTP Server powered by Apache V9 such as:

- ▶ Basic SNI support
- ▶ Trace capability
- ▶ Improved log handling capability
- ▶ Support for use of Macros in conf file
- ▶ Conditional control
- ▶ Complex authorization logic
- ▶ Support for LUA

Changed information

- ▶ Define a protected user ID for the started task
- ▶ Document comparing DGW and IHS use of directives
- ▶ Clean up of Parmlib
- ▶ Include PHP CGI programs in V9
- ▶ Define a protected user ID for the started task
- ▶ Document comparing DGW and IHS use of directives



Introduction to IBM HTTP Server for z/OS

Users of z/OS for the past several years had a choice of two HTTP Servers that they can use.

The original HTTP Server for use on z/OS (introduced in the 1990s) often was referred to as the Domino Go Webserver, DGW, or simply IHS.

When the WebSphere Application Server product became available on z/OS around 2003, it included an HTTP Server that was based on the widely used Apache HTTP Server. This version became the strategic HTTP Server on z/OS, and the original version at some point will no longer be supported.¹

The aim of this IBM Redpaper publication is to describe various features that are available in IBM HTTP Server powered by Apache, and to compare IBM HTTP Server powered by Apache with IBM HTTP Server powered by Domino. It also provides advice on how to migrate from the old version to the new version.

This chapter includes the following topics:

- ▶ 1.1, “Why you should migrate” on page 2
- ▶ 1.2, “New features in V8.5.5” on page 4
- ▶ 1.3, “New features in V9.0.0” on page 5
- ▶ 1.4, “Determining which IBM HTTP Server you are running” on page 6
- ▶ 1.5, “Checking your IBM HTTP Server version” on page 10

¹ At the time of writing, z/OS V2.1 is planned as the last supported release for DGW.

1.1 Why you should migrate

The latest IBM HTTP Server is based on Apache and it is referred to as IBM HTTP Server powered by Apache. This HTTP Server product is for z/OS that IBM is investing in for future development.

The older IBM HTTP Server, powered by Domino, was functionally stabilized for several years. The final version is IBM HTTP Server for z/OS V5.3. This product was referred to as the IBM Lotus® Domino Go Web Server (DGW). IBM announced that z/OS V2.1 is the last release of z/OS to include IBM HTTP Server powered by Domino. For more information, see the z/OS 2.1 IBM Knowledge Center at this website:

<https://ibm.biz/BdrTLs>

IBM HTTP Server powered by Apache was available for many years and is the supported IHS. It is available in z/OS Ported Tools². IHS powered by Apache supports IPv6 and 64-bit execution, and includes security authentication and authorization capabilities that are similar to those capabilities that are provided in IHS powered by Domino.

IBM now ships IBM HTTP Server powered by Apache V9 with z/OS 2.2, as described in 3.1, “IHS code that is shipped with z/OS 2.2” on page 30.

IBM recommends that clients who use IBM HTTP Server powered by Domino migrate to IBM HTTP Server powered by Apache. This migration enables you to use the capability that is provided by the Apache-based server, along with any other features that IBM might add in future releases.

For those clients who are using an IBM product that uses the Domino powered server, be aware that IBM is working to upgrade these products to replace the use of IBM HTTP Server powered by Domino with IBM HTTP Server powered by Apache. Look for documentation about each product as those changes are made or contact that product team for current information about HTTP Server support.

IBM HTTP Server is based on Apache HTTP Server 2.2.8, with more fixes. The original Apache server was developed in 1995 and is now developed and maintained by the Apache Software Foundation, which is an open source community. Apache is a C language implementation of an HTTP web server. It is widely used on many operating systems and features a wide user community. You can extend the core capability by developing your own modules or by using modules that are developed.

This wide user base provides a large community where you can obtain advice and examples of how to the Apache HTTP Server is used, which can be of use for use of IBM HTTP Server based on Apache.

IBM HTTP Server powered by Apache supports IPV6, whereas the older product does not provide this support.

² <http://www.ibm.com/systems/z/os/zos/features/unix/ported/ihs/ihsv85.html>

1.1.1 HTTP Server terminology

As described in this paper, an older and a newer IBM HTTP Server are available. It is important to establish the terminology for these products and how we refer to the products. Table 1-1 shows terminology for the two HTTP Server products.

Table 1-1 IBM HTTP Server terminology

Official Name	Short name	Latest version	How obtained
IBM HTTP Server Powered by Domino	IHS Domino or IHS DGW or DGW	V5.3	Included in z/OS until V2.1 (then discontinued)
IBM HTTP Server Powered by Apache	IHS Apache or IHS	V8.5.5	z/OS Ported Tools WebSphere z/OS
IBM HTTP Server Powered by Apache	IHS Apache or IHS	V9.0	Shipped with z/OS 2.2

On occasion, the generic terms *IBM HTTP Server* or *IHS* are used to refer to both products when the topic being described is applicable to both.

1.1.2 IBM statement of support

For more information about which versions of the IBM HTTP Server powered by Apache will be supported with z/OS 2.2, see this website:

<https://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10857>

Note: You can use your IBM Ported Tools V1.3 IHSA feature (8.5.5) concurrently with z/OS V2.2 IHSA (9.0) for some time until z/OS V2.1 is end of service (planned for September 2018).

1.1.3 Documentation

For more information about documentation for the IHS powered by Domino shipped with z/OS V1R13, see this website:

<https://ibm.biz/BdrT9U>

For more information about documentation for the IHS powered by Apache documentation at the V8.5 level, see this website:

<https://ibm.biz/BdrT9N>

For more information about documentation for the IHS powered by Apache documentation at the V9.0 level, see this website:

<https://ibm.biz/BdrT97>

For more information about migration documentation, see this website:

<https://ibm.biz/BdrTCc>

1.2 New features in V8.5.5

The latest version of IBM HTTP Server powered by Apache is V8.5.5 and includes several new features. These new features are listed in Table 1-2.

Table 1-2 New features

Feature	For more information, see...	Description
Scalability improvements (Event MPM)	Chapter 6, "Scalability and workload management" on page 103	Use of the event MPM improves the performance of the server.
z/OS Workload Management classification of requests	Chapter 6 "V8.5.5 Support for WLM"	Allows requests to be classified to WLM transaction classes.
Systems Management Facilities (SMF) logging	Chapter 8, "System Management Facilities support in IHS V8.5.5" on page 131	Logging to SMF records of servers usage information and individual requests.
z/OS operator commands	4.2.4, "Modifying command support in V8.5.5" on page 68	Provides support to allow standard type commands to be issued.
Federal Information Processing Standard (FIPS140-2) support	2.1.3, "Federal Information Processing Standards (FIPS140-2) support" on page 16	Configures System SSL to use only FIPS certified security module.
IBM MVS™ data set support	2.1.1, "Listing MVS data sets" on page 14	Allows serving of data sets without the need to use CGI.
HTTP response translation improvements	2.1.2, "HTTP response translation improvements" on page 15	Allows Content-Encoding header to influence translation.
31-bit runtime support	2.1.4, "31-bit support" on page 17	Assists with migration from IBM HTTP Server powered by Domino.

1.2.1 Updates to this IBM Redpaper publication

The updates that were made in this edition of the Redpaper publication from December 2014 are listed in Table 1-3.

Table 1-3 Updates to the Second Edition

Update	Description
"Using Health Check" on page 10	Describes how to use Health Check to check for presence of IBM HTTP Servers that are powered by Domino.
2.2, "Support for zEnterprise Data Compression" on page 17	Describes support for the IBM zEnterprise® Data Compression capability.
3.3, "Ordering and installing by using Shopz" on page 31	Describes ordering and installing the product from Shopz.
"Length of STC name consideration" on page 69	Clarifies modifying commands and Started Task name length.

Update	Description
4.6.4, "Server status by using the modify command" on page 77	Describes other modify commands that are used to display server status information.
5.3, "Migrating Library Server" on page 97	Shows how to support Library Server in an IBM HTTP Server powered by Apache.
11.1.1, "Popularity of Apache modules" on page 154	Provides more advice about module development.
"MaxSpareThreads" on page 110	Defines the MaxSpareThreads parameter.

1.3 New features in V9.0.0

The latest version of IBM HTTP Server powered by Apache is V9.0.0 and includes several new features. These new features are listed in Table 1-4. These updates that were made to this edition of the Redpaper publication were published in 2016.

Table 1-4 V9.0.0.0 list of new features

Feature	For more information, see...	Description
Basic SNI support	7.4.6, "Basic SNI Support" on page 126	Allows server to send different server-side certificate on same TCP/IP port, depending on DNS name that is used.
Trace capability	4.10, "Tracing" on page 80	Traces request processing.
Improved log handling capability	4.11, "Handling logging" on page 82	Provides improved capability in the way log files can be handled.
Support for use of Macros in configuration file	4.11, "Handling logging" on page 82	Enables the use of macros to simplify configuring a configuration file.
Conditional control	4.13, "Conditional controls" on page 83	New directives to control the directives that are used, which is an alternative to the use of ReWrite directives.
Complex authorization logic	7.2.6, "Complex authorization logic" on page 122	Enables configuring complex authorization logic to control access.
Support for LUA	12.7, "Lua support" on page 186	A powerful embeddable scripting language that is a better way to perform CGI-type programming.

Other updates made to this Redbooks publication in June 2016 are listed in Table 1-5.

Table 1-5 Other updates made in 2016

Update	Description
3.5.3, "Defining a protected user ID for the started task" on page 55	Guidance about setting up a different user ID under which to run the started tasks.
3.7, "Maintenance upgrade" on page 60	Discussion on how to plan applying maintenance and use of new new_install_root shell
5.2.15, "Comparing DGW and IHS use of directives" on page 97	IBM document that compares how to perform the same logical function in DGW and IHS.
5.2.16, "Cleaning up PARMLIB" on page 97	Removing references to DGW from PARMLIB.
12.5, "PHP CGI programs in V9" on page 174	Describes how to run PHP CGI programs, including use of authentication.

1.4 Determining which IBM HTTP Server you are running

Most z/OS clients know whether they are running an HTTP Server on z/OS and whether they are using IBM HTTP Server DGW, IBM HTTP Server Powered by Apache, or both. However, if this information is unknown, you can use methods that are described in this section to determine which server is being used.

1.4.1 Using SDSF to find running HTTP Servers

You can use SDSF to find if there are any HTTP Servers running on your z/OS LPARs.

Finding IBM HTTP Servers powered by Domino

Complete the following steps:

1. In SDSF, issue the **pre *** command so that you can see all of the STCs that are running on the z/OS LPAR.
2. Enter a **PS** command to display the SDSF process display. Look for the column that is named "Command". You might need to scroll the display to the right to find the column.
3. Issue the following **sort** command to sort the Command column display in descending order:

```
sort command d
```
4. Examine the Command column for instances of the word **IMWHTTPD**. If you find any instances, these tasks are started tasks that are running IBM HTTP Server powered by Domino. You can select that entry to view the JCL and determine which proclib it is read from and then track down the owner.

We used this approach on the LPAR that we used for this Redpaper publication. Our SDSF showed the output that is shown in Example 1-1 on page 7. Two servers that are named **IHSDC001** and **IHSDE001** are running IBM HTTP Server powered by Domino.

Example 1-1 SDSF output showing running IBM HTTP Servers powered by Domino

JOBNAME	PID	PPID	ASID	ASIDX	LatchWaitPID	Command
REXECD	131125	1	87	0057		RSHD
PMAP	33685558	1	82	0052		PORTMAP
IHSDC001	84018040	1	113	0071		IMWHTTDP
IHSDE001	84020252	1	133	0085		IMWHTTDP
IHV	131173	1	78	004E		IHVINIT
JES2S001	16908334	1	30	001E		IAZNJTCP

Finding IBM HTTP Servers powered by Apache

There is a similar process to determine whether there are any IBM HTTP Servers powered by Apache running on a z/OS LPAR. Complete the following steps:

1. In SDSF, issue the **pre *** command.
2. Issue a **PS** command to display the SDSF process display. Look for the column that is named Command. You might need to scroll the display to the right to find it.
3. Sort the display in descending order in the Command column by using the following command:

```
sort command d
```

Depending on the directory that IBM HTTP Server powered by Apache is in, it might not be easy to find a match in the display. Example 1-2 shows the output from SDSF on our z/OS LPAR and the entries that are for IBM HTTP Server powered by Apache. Although the displayed value in the Command column is truncated, we can see the string `apach`, which is enough information to determine that these started tasks are running Apache. By selecting the entry, we view the STC JCL and help find the owner.

Example 1-2 SDSF output showing running IBM HTTP Servers powered by Apache

JOBNAME	PID	Command
IHSAC001		-sh -c /ihsconfig/ihs/ihsac001/bin/apach
IHSAE002		-sh -c /ihsconfig/ihs/ihsae002/bin/apach
IHSAM001		/bin/sh -c /ihsconfig/ihs/ihsam001/bin/r
IHSAC001		/bin/sh /ihsconfig/ihs/ihsac001/bin/apac
IHSAE002		/bin/sh /ihsconfig/ihs/ihsae002/bin/apac
IHSAC001		/ihsconfig/ihs/ihsac001/bin/httpd -d /ih
IHSAC001		/ihsconfig/ihs/ihsac001/bin/httpd -d /ih

1.4.2 Determining which TCP/IP ports are used

It is useful to determine on which TCP/IP ports the HTTP Servers are listening.

One approach is to use the **netstat** command in the UNIX Systems Services (z/OS UNIX) environment of z/OS. You can access the z/OS UNIX environment by one of the following means:

- ▶ Log on to z/OS by using Telnet or SSH
- ▶ From ISPF, enter the TSO OMVS command

After you are in the z/OS UNIX environment, you can enter the **netstat** command.

Because we knew that we had an HTTP Server running that was named IHSAE002, we issued the **netstat** command and then sent that output to the **grep** command so that only lines with the string IHSAE002 are displayed. The output that was produced is shown in Example 1-3.

Example 1-3 Issuing netstat command

```
EDMCAR @ SC55:/Z1DRC1/usr/lpp/internet/bin>netstat | grep IHSAE002
IHSAE002 00AA7A07 9.12.4.28..21451      9.12.4.28..19067      Establish
IHSAE002 008F1959 0.0.0.0..8235      0.0.0.0..0           Listen
IHSAE002 00AA7A05 9.12.4.29..8235      9.190.237.133..62941  Establish
```

The line that includes the word “Listen” shows the TCP/IP port on which the IHSAE002 server is listening, which in Example 1-3 is port 8235.

The last line shows that there is a TCP/IP connection between the HTTP Server and a client. The client has a TCP/IP address of 9.190.237.133, and the host z/OS LPAR has a TCP/IP address of 9.12.4.29.

1.4.3 Accessing the home page

By using this information, you can construct the URL to use to access the home page of the HTTP Server, which in this case is the following URL:

`http://9.12.4.29:8235`

1.4.4 Using the ps command to check for HTTP Servers

Another way to check for running HTTP servers on your z/OS LPARs is to use the **ps** command from the z/OS UNIX environment. To do perform this check, you must be authorized to list all processes that are running.

Although you can use a user ID that has access to the BPX.SUPERUSER IBM RACF® rule, a safer approach is use a user ID that has access to a RACF profile that specifically grants the user the authority to perform only this requirement.

On our z/OS system, we issued the commands that are shown in Example 1-4 to give our user ID EDMCAR read access to the required RACF profile.

Example 1-4 RACF commands to allow ps command to display all processes

```
permit SUPERUSER.PROCESS.GETPSENT class(unixpriv) id(edmcar) access(read)
SETOPTS RACLIST(unixpriv) REFRESH
```

We then logged on to the z/OS LPAR by using telnet and issued the **ps -ef** command. All processes that are running in the z/OS UNIX environment were displayed. We then issued the command that us shown in Example 1-5 to find any processes that were running IBM HTTP Server for Domino.

Example 1-5 Using ps command to find running IBM HTTP Server powered by Domino

```
EDMCAR @ SC55:/u/edmcar>ps -ef | grep HTTP
IHSDC001 84018040      1 - Jun 19 ?      1:17 IMWHTTPD
IHSDC001 84020252      1 - Jun 24 ?      0:28 IMWHTTPD
```

. IHSAM001	15	2013/06/14
. IHSDC001	12	2013/06/14
. IHSDD001	12	2013/06/14
. IHSDE00#	19	2013/06/14
. IHSDE001	12	2013/06/13
. IHSDM001	12	2013/06/14

If the output shows `PARM='SH &DIR/bin/apachectl`, you are running an IBM HTTP Server Powered by Apache, as shown in Example 1-9.

Example 1-9 JCL showing use of apachectl

```
//IHS      EXEC PGM=BPXBATCH,
// PARM='SH &DIR/bin/apachectl -k &ACTION -f &CONF -DNO_DETACH',
```

If the output is similar to the output that is shown in Example 1-10 (which shows the line `EXEC PGM=IMWHTTPD`), you are running a Domino Go server.

Example 1-10 JCL showing use of IMWHTTPD

```
//WEBSRV   EXEC PGM=IMWHTTPD,REGION=OK,TIME=NOLIMIT,
// PARM=('&LEPARM/&P1 &P2 &P3')
```

1.5 Checking your IBM HTTP Server version

It is useful to determine what version of HTTP Server you are using.

1.5.1 Determining IBM HTTP Server powered by Domino version

In SDSF, select the started task that is the running IBM HTTP Server powered by Domino. The initial content of the OUTPUT DD of our running server is shown in Example 1-11.

Example 1-11 Output showing version for IBM HTTP Server powered by Domino

```
This is IBM HTTP Server V5R3M0
Built on Mar 28 2013 at 03:04:26.
Started at Mon Jun 24 20:06:23 2013
Running as "IHSDE001", UID:35002, GID:36000.
Using _CEE_ENVFILE /usr/lpp/internet/etc/httpd.envvars.
Using configuration file /etc/ihsde001/etc/httpd.conf.
```

As shown in Example 1-11, the version is V5R3M0.

Using Health Check

IBM developed a Rexx program that can be used as part of the Health Check capability to check for the presence of IBM HTTP Servers powered by Domino. It can be used to help prepare for the migration to z/OS 2.2. For more information about updates to the migration Health Check about this Rexx, see the following z/OS migration and installation website:

<http://www.ibm.com/systems/z/os/zos/installation/>

Updating SAXREXX

We obtained an initial version of this REXX from the IBM developer and copied it to a data set that is concatenated to our SAXREXEC. We were advised to not copy it into the SYS1.SAXREEXEC data set. On our system, we looked in the SYS1.PARMLIB(AXR00) member and saw that a data set that is named WTSCPLX1.REXXEXEC was added to the system REXX setup. We put the REXX into that data set.

Update HZSPARM

Copy the text that is shown in Example 1-12 into HZSPRMxx of your SYS1.PARMLIB.

Example 1-12 Text to update HXSPARM

```
ADDRP CHECK(IBMZMIG,ZOSMIG_HTTP_SERVER_DOMINO_CHECK)
      EXEC(DOMCHK)
      REXXHLQ(IBMZMIG)
      REXXTSO(YES)
      REXXIN(NO)
      MSGTBL(*NONE)
      USS(NO)
      VERBOSE(NO)
      SEVERITY(MEDIUM)
      INTERVAL(168:00)      <=== once a week
      ACTIVE                <=== change to inactive, if you don't want it to run right
now.
      EINTERVAL(SYSTEM)
      DATE(20140915)
      PARM('')
      REASON('Verify that IBM HTTP Server Domino is not in use.')
```

Importing change dynamically

Issue commands similar to the following commands to dynamically activate the entries that are shown in Example 1-12:

```
F HZSPROC,ADD,PARMLIB=(xx)
F HZSPROC,REPLACE,PARMLIB=(xx,yy)
```

Example result

We started a IBM HTTP Server powered by Domino on our z/OS LPAR and then issued the command. In the z/OS system log, we saw the messages that are shown in Example 1-13.

Example 1-13 Messages from running Health Check

```
F HZSPROC,ADD,PARMLIB=(02)
IEE252I MEMBER HZSPRM02 FOUND IN SYS1.PARMLIB
HZS0400I CHECK(IBMZMIG,ZOSMIG_HTTP_SERVER_DOMINO_CHECK): 769
ADD PROCESSING HAS BEEN COMPLETED
HZS0403I ADD PARMLIB PROCESSING HAS BEEN COMPLETED
D OMVS,ASID=ALL
BPX0070I 04.03.22 DISPLAY OMVS 772
OMVS      0011 ACTIVE          OMVS=(1A)
USER      JOBNAM  ASID      PID      PPID STATE   START      CT_SECS
OMVSKERN  BPX0INIT 002C      1        0 MR----- 08.46.23    22.8
      LATCHWAITPID=      0 CMD=BPXPINPR
      SERVER=Init Process      AF=      0 MF=00000 TYPE=FILE
STC      RESOLVER 0015      131074      1 1R---B-- 08.46.23    1.7
      LATCHWAITPID=      0 CMD=EZBREINI
IHSDE001  IHSDE001 0025      33686057      1 HK----- 04.02.44     .0
      LATCHWAITPID=      0 CMD=IMWHTTPD
```

```
HZS0002E CHECK(IBMZMIG,ZOSMIG_HTTP_SERVER_DOMINO_CHECK): 773
DOMCHK8 One or more IBM HTTP Server(s) Powered by Domino were found.
IHSDE001
```

The last message that is shown identified the started task IHSDE001 as being an IBM HTTP Server powered by Domino.

1.5.2 Determining IBM HTTP Server powered by Apache version

The started task that runs IBM HTTP Server powered by Apache does not display any version information. This information often can be found in the error log file. Messages that include the version are written to this log file when it is started, as shown in Example 1-14.

Example 1-14 Output showing version for IBM HTTP Server powered by Apache

```
Using config file /ihsconfig/ihs/ihsae001/conf/httpd.conf with -DNO_DETACH -DZOS
IBM_HTTP_Server/8.5.5.1 (UNIX) configured -- resuming normal operations
```

As shown in Example 1-14, the version of V8.5.5.1. Another option is to use the **apachectl** command, as shown in Example 1-15. This approach shows the version and build date.

Example 1-15 Using apachectl to get version and build date

```
EDMCAR @ SC55:/ihsconfig/ihs/ihsae001/bin>./apachectl -v
Server version: IBM_HTTP_Server/8.5.5.1 (UNIX)
Server built:   May 23 2013 00:51:38
```

1.5.3 For more information

For more information about how to find the IBM HTTP Server version you are running, see this website:

<https://ibm.biz/Bdrkfv>



Features and performance

This chapter describes some of the new features in V8.5.5 of IBM HTTP Server powered by Apache that are not covered elsewhere in this publication. Also described is a 2006 performance comparison of the two IBM HTTP Servers that are available on z/OS.

This chapter includes the following topics:

- ▶ 2.1, “New features in V8.5.5” on page 14
- ▶ 2.2, “Support for zEnterprise Data Compression” on page 17
- ▶ 2.3, “Functional differences” on page 23
- ▶ 2.4, “Performance comparison” on page 24

2.1 New features in V8.5.5

In the following sections, we describe the following new features that are not covered elsewhere in this paper:

- ▶ 2.1.1, “Listing MVS data sets” on page 14
- ▶ 2.1.2, “HTTP response translation improvements” on page 15
- ▶ 2.1.3, “Federal Information Processing Standards (FIPS140-2) support” on page 16
- ▶ 2.1.4, “31-bit support” on page 17
- ▶ 2.2, “Support for zEnterprise Data Compression” on page 17

2.1.1 Listing MVS data sets

IBM HTTP Server powered by Domino supplied a sample GWAPI that provided a way to allow users view z/OS data sets from a browser. Before Version 8.5.5, IBM HTTP Server powered by Apache supplied a sample CGI program that provided a similar capability. For more information, see this website:

<http://people.apache.org/~gregames/mvsds>

Version 8.5.5, IBM HTTP Server powered by Apache supplies a new module that includes a more integrated way of providing this capability that does not use CGI programs. To use this module, you must add the following line to your `httpd.conf` file:

```
LoadModule mvsds_module modules/mod_mvsds.so
```

Then, you add new directives to the `httpd.conf` so that the server can display z/OS data sets, as shown in Example 2-1.

Example 2-1 Adding directives to allow viewing of z/OS data sets

```
<VirtualHost wtsc55.itso.ibm.com:8235>
<Location /mvsds >
# Treat URL's as dataset names
MVSDFS ON
#
# Data sets often lack file extensions
DefaultType text/plain
#
# Allow PDS listings
MVSDFSIndexes On
</Location>
```

We added the directives that are shown in Example 2-1 to a server on our system and used the `http://wtsc55.itso.ibm.com:8235/mvsds/'edmcarr.z.cntl'` to view a data set.

The produced output is shown in Example 2-2.

Example 2-2 Listing of data set members as seen in browser

Directory Listing of 'EDMCAR.Z.CNTL':

Name	Size	Created	Changed		ID
APPLYPTF	11	2009/03/04	2009/03/04	01:54:35	EDMCAR
ASMEXIT	23	2008/06/12	2008/06/12	22:56:07	EDMCAR
ASMRAUTH	23	2013/03/13	2013/04/08	05:07:42	EDMCAR
ASMRMMGR	22	2012/11/11	2013/02/28	19:27:48	EDMCAR
BPXBATCH	11	2010/02/16	2010/04/28	22:53:51	EDMCAR

We then clicked a member and its content was displayed.

This feature cannot display a list of data sets that match a specific mask. For example, the following URL does not work:

`http://wtsc55.itso.ibm.com:8235/mvsds/'edmcar'`

You must enter the full name of the data set that you want to view.

2.1.2 HTTP response translation improvements

Traditionally, IBM HTTP Server powered by Apache performs translation between characters sets that are based on the value of the Content-Type header alone. In Version 8.5.5, this translation can now be influenced by using the Content-Encoding header. This header might be useful in CGI programs that are sending data in EBCDIC or ASCII and want to influence how the server translates. This feature is enabled by adding the following directive:

```
CharsetOptions DGWCompat
```

This directive can be added to affect all requests or added within specific Location directives, as required. We set up a simple Rexx program as a test. Our code is shown in Example 2-3.

Example 2-3 Rexx CGI program to demonstrate DGWCompat

```
/* REXX */
say 'Content-type: text/html;charset=UTF-8'
say 'Content-Encoding: EBCDIC'
/* This next line separates the HTTP Header above from the
   the HTTP response body, and must be present */
say ''
say 'Hello from a Rexx CGI at time: 'time()
asciiString = '30313233343536373839'x
ebcdicString = 'F0F1F2F3F4F5F6F7F8F9'x
say 'Ascii numbers: ' asciiString '<br>'
say 'Ebcdic numbers: ' ebcdicString '<br>'
exit
```

When we ran this CGI program, we saw the output that is shown Example 2-4.

Example 2-4 Output when encoding header set to EBCDIC

```
Hello from a Rexx CGI at time: 21:34:31 Ascii numbers: ????????  
Ebcidic numbers: 0123456789
```

We then changed the following line of the Rexx:

```
say 'Content-Encoding: EBCDIC'
```

to

```
say 'Content-Encoding: ASCII'
```

Rerunning the request produced the output that is shown in Example 2-5.

Example 2-5 Output when Encoding header is ASCII

```
è???@???@?@?@???@???@???z@??z??z?-????@???????z@@0123456789  
@L??n-????@???????z@@????????@L??n-  
L
```

Most of the output was now unreadable, but the ASCII string of numbers that was previously unreadable is now readable.

2.1.3 Federal Information Processing Standards (FIPS140-2) support

Support is provided for the Federal Information Processing Standard (FIPS140-2).

The FIPS 140-2 standard was published by the National Institute of Standards and Technology (NIST).

This standard defines the security requirements that must be satisfied by a cryptographic module that is used in a security system that is protecting unclassified information within IT systems. It is intended to cover various potential applications and environments in which cryptographic modules might be deployed.

FIPS140-2 support is enabled by using the SSLFIPSEnable directive, which can be used only in global scope in the httpd.conf file on z/OS. We configured System SSL to use only a FIPS-certified security module. Restarting the server cannot be used to pick up changes that are related to this directive; instead, you must stop and then restart the server.

An example of how to use the SSLFIPSEnable directive is shown in Example 2-6.

Example 2-6 Example of using the SSLFIPSEnable directive

```
# z/OS: Global Scope only.  
SSLFIPSEnable  
<VirtualHost *:443>  
SSLEnable  
KeyFile safkeyring:///WASKeyring  
</VirtualHost>
```

2.1.4 31-bit support

Version 8.5.5 of IBM HTTP Server powered by Apache now includes a 31-bit run time to help migrate from DGW.

This feature can be useful if you have a GWAPI in IBM HTTP Server powered by Domino, which used a product that provided 31-bit libraries only. If you need to convert the GWAPI to an Apache style module, you must create a server that runs in 31-bit mode so that the modules can call the 31-bit libraries.

This feature also was provided to allow IBM staff who currently use IBM HTTP Server powered by Domino and rely on 31-bit libraries to plan to migrate to IBM HTTP Server powered by Apache.

To set up a server that runs in 31-bit mode, you use the `install_ihs` script that is in the `.31bit` subdirectory.

2.1.5 Features in IHS powered by Domino and not in IHS powered by Apache

IBM HTTP Server DGW supports the following features:

- ▶ Go Webserver Application Programming Interface (GWAPI). If you have any applications, you must rewrite the modules as described in 2.1.4, “31-bit support” on page 17.
- ▶ Fast Response Cache Accelerator (FRCA).

2.2 Support for zEnterprise Data Compression

The zEnterprise Data Compression capability provides an efficient way of compressing data. It is implemented in hardware that is available for System z. This capability uses significantly less CPU than equivalent software compression methods.

APAR PI24424 supports the IBM HTTP Server powered by Apache to use the zEnterprise Data Compression capability. For more information about this APAR, see this website:

<http://www.ibm.com/support/docview.wss?uid=isg1PI24424>

In our small scale testing, we saw CPU savings of approximately 85%.

If you are using your IBM HTTP Server powered by Apache to deliver large response files that are good candidates for compression, use of this capability can significantly reduce the required CPU.

The APAR delivers a new module to use in place of the default compression module. The default compression module is `mod_deflate.so`.

The module that is supplied by this APAR is called `mod_deflate_z.so`.

If the size of the response is relatively small or there are dynamic responses that start with small flushed chunks, the data can be compressed in software for efficiency reasons.

2.2.1 zEnterprise Data Compression requirements

To use zEnterprise Data Compression, the following requirements must be met:

- ▶ Use a zEC12 or BC12
- ▶ Peripheral Component Interconnect Express (PCIe) hardware adapter is installed
- ▶ Running z/OS 2.1
- ▶ Enabled the feature by using the IFAPRDxx member in SYS1.PARMLIB

Note: The Peripheral Component Interconnect Express (PCIe) hardware adapter and the associated software are a chargeable feature.

On the z/OS system that was used to test this capability, the IFAPRD01 member contained the following control statement:

```
WHEN (SYSNAME(*)) PRODUCT NAME(*) STATE(ENABLED)
```

2.2.2 Verifying that zEnterprise Data Compression is active

To verify that the zEnterprise Data Compression capability is available on a z/OS LPAR, issue a **D PCIE** command. Example 2-7 shows the output that is generated from the use of this command on the z/OS LPAR that we tested. It includes output from issuing a more specific command to get more information about an individual accelerator.

Example 2-7 Example output from D PCIE command

```
D PCIE
IQP022I 23.47.03 DISPLAY PCIE 033
PCIE      0012 ACTIVE
PFID  DEVICE TYPE NAME      STATUS  ASID  JOBNAME  PCHID VFN
0033  Hardware Accelerator   ALLC   0013  FPGHWAM  05D0  0004
0023  Hardware Accelerator   ALLC   0013  FPGHWAM  0578  0004

D PCIE,PFID=33
IQP024I 23.47.19 DISPLAY PCIE 035
PCIE      0012 ACTIVE
PFID  DEVICE TYPE NAME      STATUS  ASID  JOBNAME  PCHID VFN
0033  Hardware Accelerator   ALLC   0013  FPGHWAM  05D0  0004
CLIENT ASIDS: NONE
Application Description: zEDC Express
Device State: Ready
Adapter Info - Relid: 00000B Arch Level: 03
                Build Date: 02/13/2014 Build Count: 03
Application Info - Relid: 000000 Arch Level: 02
```

FPGHWAM is an address space that is automatically started during z/OS initialization if the PCIe facilities hardware is installed.

2.2.3 Enabling use of zEnterprise Data Compression

To enable an IBM HTTP Server powered by Apache to use the zEnterprise Data Compression, replace the following line in the `httpd.conf`:

```
LoadModule deflate_module modules/mod_deflate.so
```

with

```
LoadModule deflate_module modules/mod_deflate_z.so
```

The server must be stopped and started or recycled to incorporate this change.

2.2.4 Testing

To test the effectiveness of zEnterprise Data Compression, we created a file in the `htdocs` subdirectory of our server. We named this file `sc63-uss.js`. This file was not an actual Java script file because it contained only a large directory listing of the UNIX System Services environment on the z/OS LPAR. The file was 23844175 bytes, or nearly 24 MB.

Example 2-8 shows how we modified the `httpd.conf` so that the deflate module is called when a request to access the `sc63-uss.js` was received by the server. We also added the three `DeflateFilterNote` directives to create three keywords that we can refer to in the `LogFormat` directive.

Example 2-8 Changes to start deflate module for Java script file

```
<IfModule mod_deflate.c>
  AddOutputFilterByType DEFLATE text/plain text/html
    <filesMatch "\.(js|css|html|mp3)$">
      SetOutputFilter DEFLATE
    </filesMatch>
  DeflateFilterNote Input instr
  DeflateFilterNote Output outstr
  DeflateFilterNote Ratio ratio
</IfModule>
```

The following original `LogFormat` was in the `httpd.conf`:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

It was modified as shown in the following example:

```
LogFormat "%h %l %u %t \"%r\" %>s %b %D %{outstr}n/%{instr}n %{ratio}n%%" common
```

The `%D` results in the time that is taken to process the request to be output. The text after `%D` outputs the size of the file after it was compressed, the size of the file before it was compressed, and the compression ratio achieved.

We used the following URL to access the `sc63-uss.js` file:

```
http://wtsc63.itso.ibm.com:8265/sc63-uss.js
```

Using the default deflate module

When we used the default Apache deflate module and accessed the file, we saw the following entry in the `access_log`:

```
GET /sc63-uss.js HTTP/1.1" 200 1728840 15919520 1728822/23844175 7%
```

Using the new deflate module

When we used the new deflate module and accessed the file, we saw the following entry in the access_log:

```
GET /sc63-uss.js HTTP/1.1" 200 2713329 28821637 2713311/23844175 11%
```

2.2.5 SMF information

We set up our server to output an SMF record for each request. We accessed the file several times for each of the deflate modules.

The details of a typical request to access the sc63-uss.js file by using the default deflate module are shown in Example 2-9.

Example 2-9 SMF record for requests processed by default deflate module

```
Record#: 7;  
  Type: 103; Size: 111; Date: Thu Nov 20 04:31:55 EST 2014;  
  SystemID: SC63; SubsystemID: STC; Flag: 94;  
  Subtype: 14 (IHS Request Info);  
pid=262356 method=GET host=wtsc63.itso.ibm.com:8265 uri=/sc63-uss.js rip =  
9.190.237.75 elapsed= 18507 cpu=0.42540
```

The details of a typical request to access the sc63-uss.js file by using the new deflate module are shown in Example 2-10.

Example 2-10 SMF record for requests processed by new deflate module

```
Record#: 16;  
  Type: 103; Size: 111; Date: Thu Nov 20 04:00:33 EST 2014;  
  SystemID: SC63; SubsystemID: STC; Flag: 94;  
  Subtype: 14 (IHS Request Info);  
pid=84148426 method=GET host=wtsc63.itso.ibm.com:8265 uri=/sc63-uss.js rip =  
9.190.237.75 elapsed= 23198 cpu=0.06992
```

2.2.6 Comparing results

The performance of typical requests that are processed by the default deflate module and the new deflate module that uses the zEnterprise Data Compression capability are compared in Table 2-1.

Table 2-1 Comparing performance of the two deflate modules

Module	Length after compression	Ratio (Percent compressed)	CPU used
Default deflate	1728822	7 (93)	0.42540
New deflate	2713311	11 (89)	0.06992

Conclusion

The results show that the new deflate module use of zEnterprise Data Compression results in significant CPU savings of approximately 87% when the two individual requests are compared.

The zEnterprise Data Compression was slightly less than the compression that was achieved by the default deflate module. However, the reduced compression is more than offset by the significant CPU savings.

2.2.7 SMF information about hardware compression

SMF records that record usage of the zEnterprise Data Compression feature are written as SMF type 74, subtype 9 records.

Extracting the SMF records

We ran the JCL that is shown in Example 2-11 to extract these SMF records after sending several requests to the server.

Example 2-11 JCL to extract relevant SMF records

```
//GETSMF EXEC PGM=IFASMFDP
//DUMPIN DD DSN=SYS1.SC63.MAN2,DISP=SHR
//DUMPOUT DD DSN=EDMCAR.SMFDATA.T74,DISP=(NEW,CATLG),
// SPACE=(CYL,(10,10),RLSE),DCB=(RECFM=VB,LRECL=32760)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
INDD(DUMPIN,OPTIONS(DUMP)),
OUTDD(DUMPOUT,TYPE(74(9)))
```

Producing RMF report

We then ran the JCL that is shown in Example 2-12 to produce an IBM RMF™ report about the usage of the zEnterprise Data Compression feature.

Example 2-12 JCL to produce an IBM RMF report

```
//RMFPP EXEC PGM=ERBRMFPP,REGION=0M
//MFPINPUT DD DSN=EDMCAR.SMFDATA.T74,DISP=SHR
//MFPMSGDS DD SYSOUT=*
//XPRPTS DD DISP=(NEW,CATLG),
// DSN=EDMCAR.RMF.PCIE.XML,
// UNIT=SYSDA,
// SPACE=(TRK,(15,10)),
// DCB=(LRECL=256,RECFM=VB,BLKSIZE=0)
//SYSIN DD *
RTOD(0000,2400)
STOD(0000,2400)
SUMMARY(INT)
REPORTS(PCIE)
SYSOUT(T)
DINTV(0002)
```

Note: The RMF report is written as XML to the XPRPTS data set.

Downloading RMF XML Toolkit for Windows

To display the XML report, you need associated style sheets. These style sheets are included as part of the RMF XML Toolkit for Windows and can be downloaded at this website:

<https://ibm.biz/BdrkvQ>

At this website, a file that is named `erbxm1tk_110.msi` is available for download. We downloaded this file, ran it, and it installed the toolkit into the `C:\zProducts\RMF\RMF Postprocessor XML Toolkit` directory that we manually selected.

Downloading the data set to PC

Use FTP with ASCII transfer to download the data set to your Windows PC.

Store the download file in the same directory in which the RMF Processor XML Toolkit was stored. In our case, we downloaded the data set to the `C:\zProducts\RMF\RMF Postprocessor XML Toolkit` directory and named it `rmf-zedc-rept.xml`.

Viewing the XML report

To view the report, we entered the following URL in our browser:

`file:///C:/zProducts/RMF/RMF Postprocessor XML Toolkit/rmf-zedc-rept.xml`

The output that was produced is shown in Figure 2-1. The output gives you information about how the zEnterprise Data Compression hardware performed.

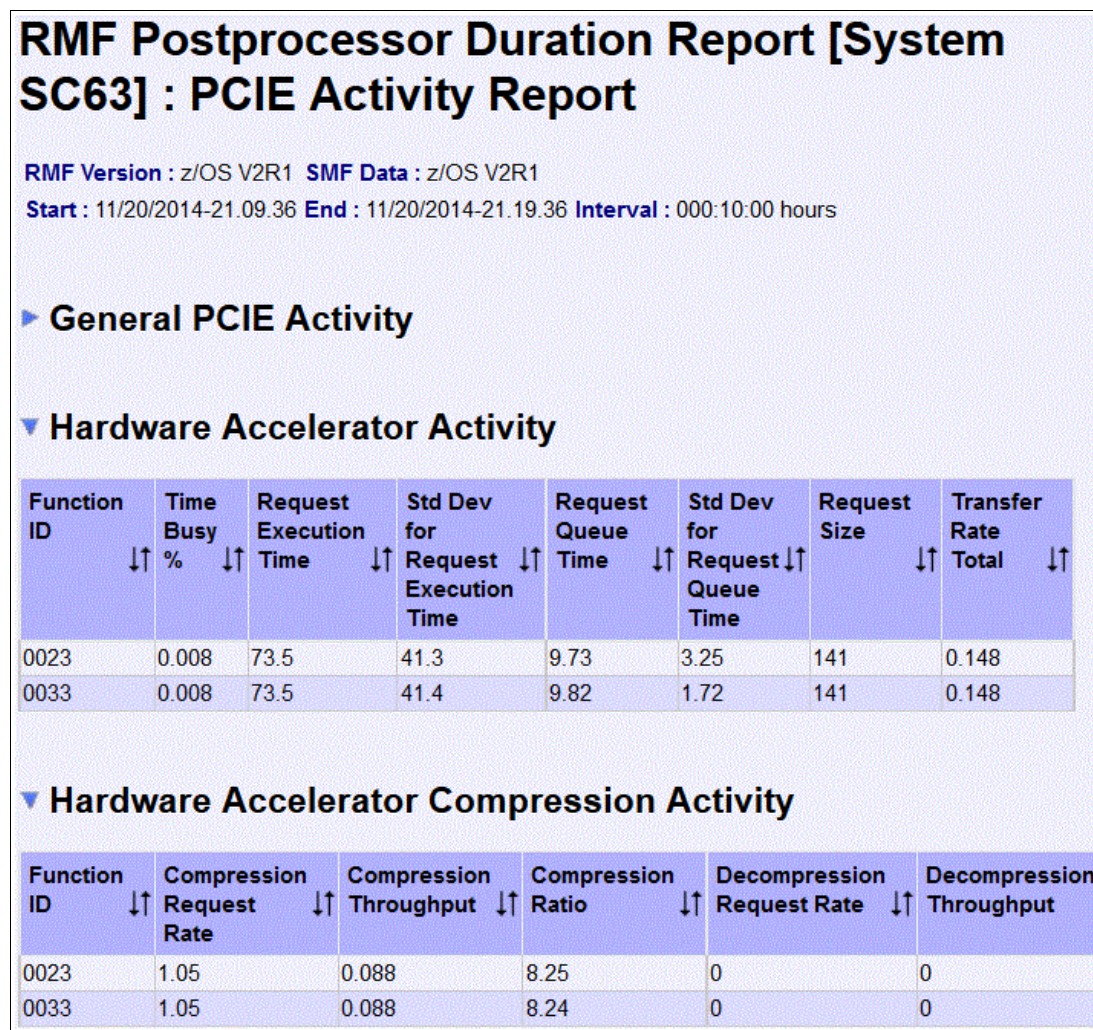


Figure 2-1 Viewing PCIE XML report in a browser

For more information about the PCIE Activity report, see the following IBM z/OS 2.1 Knowledge Center website:

<https://ibm.biz/Bdrkvw>

2.2.8 Compression log usage information

An APAR is included as part of the V8.5.5.5 release of the IBM HTTP Server powered by Apache with which you can log compression information about each request. For more information about APAR PI30041, see this website:

<http://www.ibm.com/support/docview.wss?uid=swg1PI30041>

How the LogFormat directive might be coded so that compression information was displayed is shown in Example 2-13. An 'H' indicates that the zEnterprise Data Compression hardware compression was used. An 'S' indicates that software compression was used, and a '-' indicates that no compression was used.

Example 2-13 Example directive and sample log output

```
LogFormat "%h %l %u %t \"%r\" %>s %b %{Content-Encoding}o %{zEDC}n" common

9.80.82.19 - - [20/Nov/2014:20:04:49 -0500] "GET /big.txt HTTP/1.1" 200 47200 gzip
H
9.80.82.19 - - [20/Nov/2014:20:06:16 -0500] "GET /medium.txt HTTP/1.1" 200 30 gzip
S
9.80.82.19 - - [20/Nov/2014:20:06:16 -0500] "GET /other.xyz HTTP/1.1" 200 20 gzip
-
```

2.3 Functional differences

IBM HTTP powered by Apache supports IPV6; IBM HTTP Server for z/OS powered by Domino does not.

IBM HTTP Server DGW and IHS powered by Apache both can be run as started tasks, but the initial program they start is different. IHS DGW starts a program that is stored in a PDS. The IHS powered by Apache starts the **apachectl** program that is stored in a directory of the z/OS UNIX environment. For more information, see 4.1, “Running IBM HTTP Server powered by Apache” on page 66.

IBM HTTP Server DGW runs with a fixed number of threads by default. IBM HTTP Server powered by Apache runs multiple processes with multiple threads at start time. More processes and threads can be added dynamically if the demand increases. For more information, see Chapter 6, “Scalability and workload management” on page 103.

Both versions of IBM HTTP Server can serve content that is stored in EBCDIC and ASCII format. For more information about setting up character set conversion in IBM HTTP Server powered by Apache, see this website:

http://publib.boulder.ibm.com/httperv/manual70/mod/mod_charset_lite.html

IBM HTTP Server DGW is installed by using the `setup.sh` script. IBM HTTP Server powered by Apache is installed by using `install_ihs`. For more information, see Chapter 3, “Installing your first IHS” on page 29.

2.4 Performance comparison

In 2006, the Washington Systems Center conducted a series of performance tests with both servers and concluded there was little performance difference between them. For more information about this report, see the following resources:

- ▶ Chapter 3, “Installing your first IHS” on page 29
- ▶ The content of the report, which is available at this website:
<http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101170>

Although many improvements were made to IBM HTTP Server powered by Apache, there were no improvements made to IBM HTTP Server powered by Domino.

2.4.1 Basic measure of throughput test

The first test shows a basic measure of throughput, as measured in MBps. Static objects were served with no caching. The number of users who are presented to the HTTP Server was scaled up from 100 to 400. The conclusion was that at this basic level of comparison of throughput, the two HTTP Servers are essentially the same across the number of users presented, as shown in Figure 2-2.

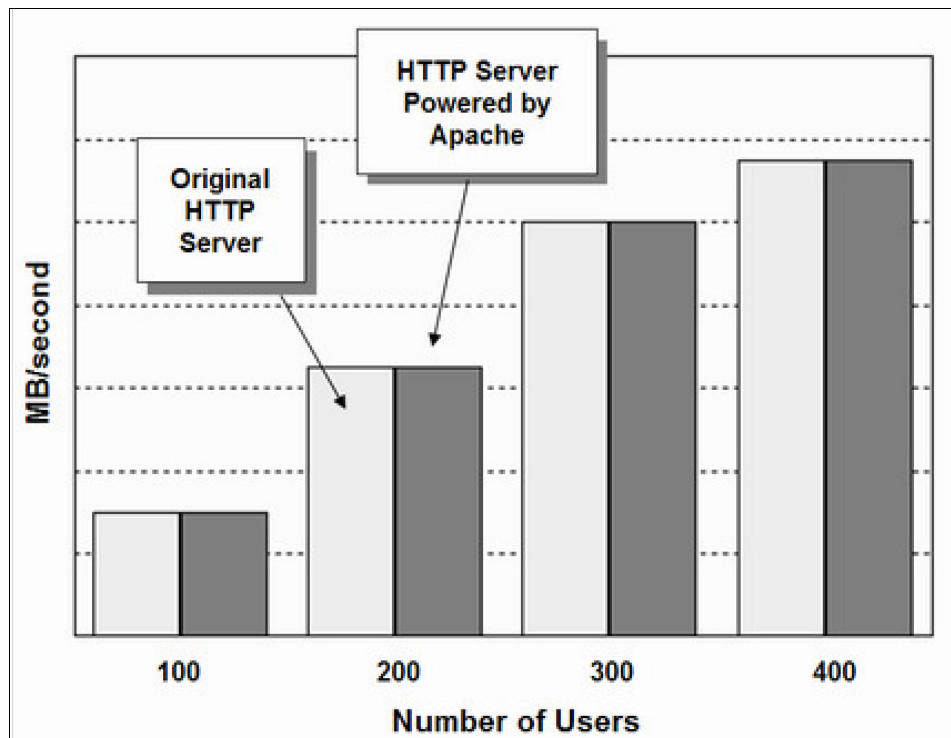


Figure 2-2 Measure of throughput

2.4.2 CPU utilization test

The second test shows the CPU utilization per 100 static pages served. Again, the objects were static and no caching was used. The number of users who are presented to the HTTP Server was scaled up from 100 to 400. We see that the original HTTP Server for z/OS has a slight CPU advantage over the HTTP Server for z/OS powered by Apache. We believe that this result is because of the longer period the original HTTP Server was available and the more extensive source code tuning that occurred, as shown in Figure 2-3.

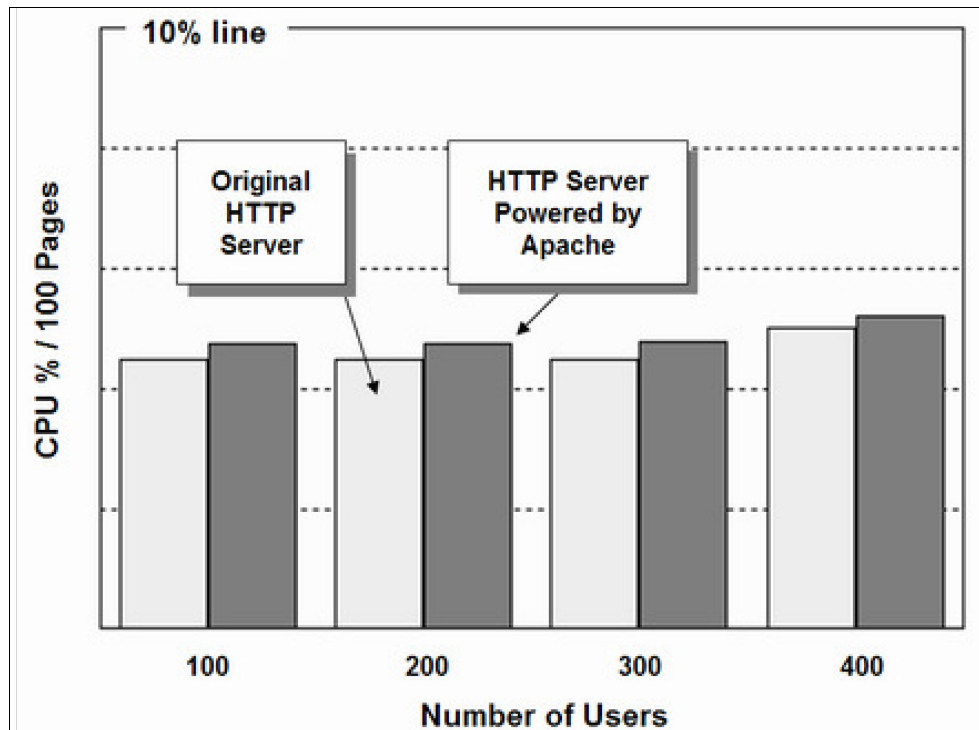


Figure 2-3 CPU utilization

2.4.3 Measure of throughput with and without caching test

The third test was a measure of throughput (measured in MBps) with and without caching. Static pages were served. The original HTTP Server used the Fast Response Caching Accelerator (FRCA) function of TCP on z/OS. The HTTP Server powered by Apache used its own internal caching, as shown in Figure 2-4.

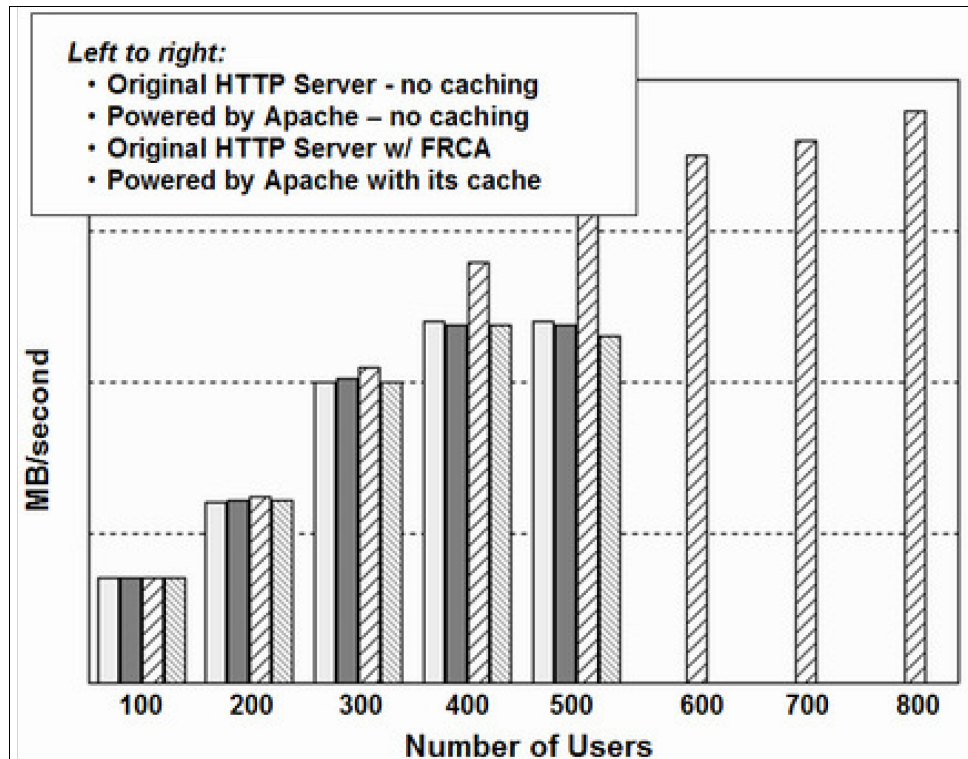


Figure 2-4 Throughput measured in MBps

2.4.4 Measure of CPU test

The final test was a measure of CPU while using the IBM WebSphere Application Server for z/OS plug-in to handle a simple transaction request. In Figure 2-5, the vertical axis shows CPU % per 100 transaction requests, and the horizontal axis shows increasing numbers of users.

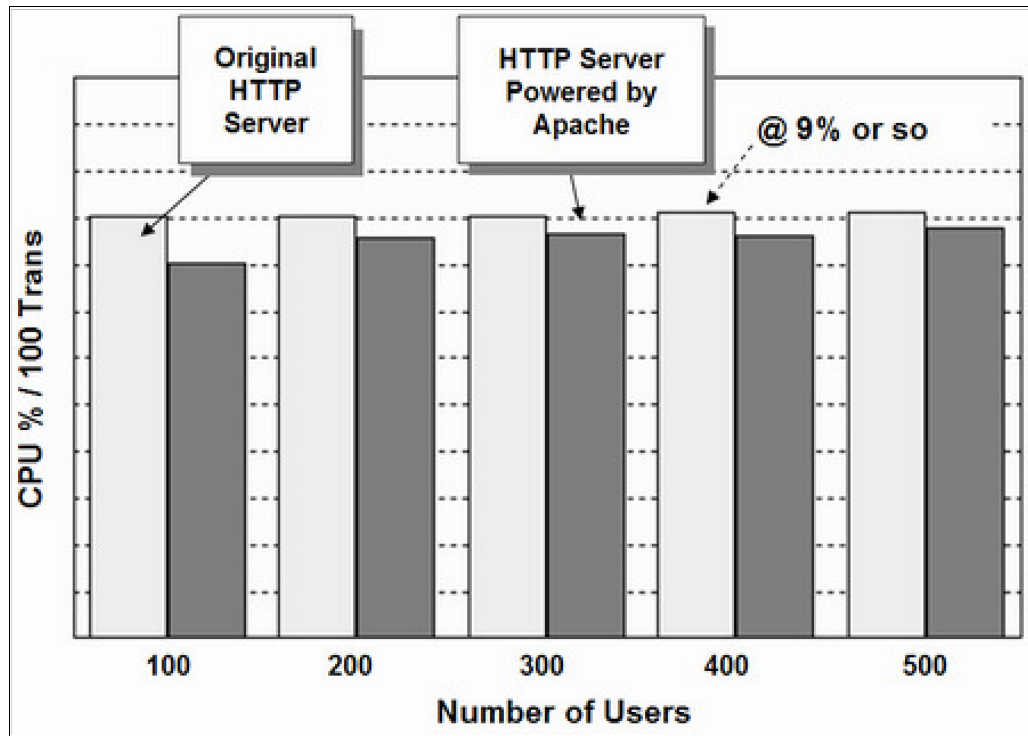


Figure 2-5 Measure of CPU that uses the IBM WebSphere Application Server plug-in

IBM HTTP Server for z/OS powered by Apache saw a moderate performance advantage compared to the original HTTP Server. The difference is approximately 9% when the CPU utilization numbers of the two are compared.



Installing your first IHS

This chapter describes the installation and setup of your first IBM HTTP Server powered by Apache.

This chapter includes the following topics:

- ▶ 3.1, “IHS code that is shipped with z/OS 2.2” on page 30
- ▶ 3.2, “Obtaining and installing the product code” on page 30
- ▶ 3.3, “Ordering and installing by using Shopz” on page 31
- ▶ 3.4, “Installation when a component of another IBM product” on page 52
- ▶ 3.5, “Sample real-world setup process” on page 53
- ▶ 3.6, “Using intermediate symbolic links” on page 58
- ▶ 3.7, “Maintenance upgrade” on page 60

3.1 IHS code that is shipped with z/OS 2.2

The product code for IBM HTTP Server powered by Apache on z/OS is shipped automatically as part of z/OS 2.2.

The following example shows a typical default location on z/OS 2.2 for the product code:

```
/usr/lpp/ihsa_zos
```

You can begin defining IHS servers without downloading any other software. For more information, see 3.5, “Sample real-world setup process” on page 53.

3.2 Obtaining and installing the product code

Before you can configure an IBM HTTP Server powered by Apache on z/OS, you must obtain and install the product code because it is not included with the z/OS operating system. Only IBM HTTP Server powered by Domino is included with z/OS.

The product code for IBM HTTP Server powered by Apache on z/OS can be obtained by using one of the following methods:

- ▶ Delivered as a component of other IBM products
- ▶ Downloaded at no charge from the IBM Shopz website

3.2.1 Delivered as a component of other IBM products

If your company purchased WebSphere Application Server for z/OS, Business Process Manager for z/OS, or Operational Decision Manager for z/OS, a compatible version of IBM HTTP Server powered by Apache product code is included with these products.

For more information about this method, see 3.4, “Installation when a component of another IBM product” on page 52.

3.2.2 Downloaded at no charge from the IBM Shopz website

If you do not have these products, you can download IBM HTTP Server powered by Apache on z/OS (via program number 5655-M23) at no charge from the following IBM Shopz website:

https://www.ibm.com/software/shopzseries/ShopzSeries_public.wss

IBM HTTP Server powered by Apache is part of the IBM z/OS Ported Tools. For more information about the z/OS Ported Tools, see this website:

<http://www.ibm.com/systems/z/os/zos/features/unix/ported>

After downloading the IBM HTTP Server powered by Apache from Shopz, use SMP/E to install the software. For more information about this process, see 3.3, “Ordering and installing by using Shopz” on page 31. This approach does not use IBM Installation Manager.

3.3 Ordering and installing by using Shopz

This section describes the process of ordering IBM HTTP Server powered by Apache from the IBM Shopz website and then installing it. The result of this process is a zFS data set that is mounted at a nominated directory in the z/OS UNIX environment on the z/OS LPAR, which contains the executable IBM HTTP Server powered by Apache product code.

At the time of this writing, the downloaded product code resulted in V8.5.5.3 when installed by using SMP/E. This release level of the product requires applying the PTF UI22400 at the same time to resolve an installation issue. Applying this PTF as part of the installation process is described in the following section.

3.3.1 IBM Shop z website

The IBM Shopz site is available at this website:

https://www.ibm.com/software/shopzseries/ShopzSeries_public.wss

The welcome window is shown in Figure 3-1.



Figure 3-1 IBM Shopz welcome window

3.3.2 Ordering software

As an IBM customer, click **Sign in for registered users** at the welcome window. A logon window opens in which you can enter your User ID and password.

For the purposes of documenting the ordering process, we logged in by using the link for IBM Employees. Although the steps that are shown in this section show this approach, the steps are the same for IBM customers. Complete the following steps:

1. After logging on, the Shopz welcome window opens. To place a new order for IBM HTTP Server powered by Apache, click **Create new software orders for services or products**.

The next window shows the beginning of the process to create an order. Select the **z/OS -Products** option.

2. From the drop-down menu for z/OS-Products, select the **CBPDO (products)** option, as shown in Figure 3-2.

Shopz > My orders > Create new order >

My orders

Create new order Draft orders In process Completed

To begin the software ordering process, select the appropriate values below. Packages indicated with an asterisk (*) require service contracts.

Customer number 994616

Package category [\[Help\]](#)

☐ z/OS - Service Individual PTFs

☒ z/OS - Products CBPDO (products)

Figure 3-2 Start of create new order process

3. Click **Continue**. The Step 1 of 8 Specify Order Basics window opens, which shows your customer number and other information. There is a field at the top of the window that is named Order name that auto-generates a value, such as Products - 2014-11-11 16.42.03. Change this value to IHS Apache - 2014-11-11 16.42.03, and click **Continue**.
4. The Step 2 of 8 Select hardware systems window opens, which lists information about your site. Select an appropriate entry. As IBM HTTP Server powered by Apache product is no charge from IBM, you order it only once for your site, not for every hardware system that appears in the list. This one copy of the software you obtain can be used on every z/OS environment you have. Click **Continue**.
5. The Step 3 of 8 Report installed software window opens. Select the **Do not use a report for this order** option and click **Continue**.

6. The Step 4 of 8 Shop for products window opens. Click **Group** and select **MVS - System Mgmt. and Security (81 products)**. In the Filter menu, select **Show all products**. In the Search field menu, select **Product description** and in the Search for field, enter HTTP, as shown in Figure 3-3.

Step 4 of 8 Shop for products

IHS Apache - 2014-11-11 16.42.03

Select a view of the product catalog to add products. Repeat the process with other views, and then continue to the next step to review the complete contents of your order.

Products in order: 0

Catalog view

Group

Language

Filter

Search field ☐ Product ID ☒ Product description

Search for

Sort by ☒ Product ID ☐ Product description

Upload an installed software report for more filtering options.

Figure 3-3 Locating the IBM HTTP Server powered by Apache product

- Click **Show catalog** to update the display to show the IBM HTTP Server powered by Apache product, which is part of the IBM Ported Tools (see Figure 3-4).

The screenshot shows a window titled "MVS: System Mgmt. and Security". At the top, there is a button labeled "Show catalog". Below this is a table with the following columns: "Select", "Product", "Description", "Version", "Language", and "Notes". The table contains one row with the following data: "Select" has a checkbox and three small icons (a triangle, a circle, and a diamond); "Product" has the value "[5655-M23]"; "Description" has the value "IBM Ported Tools: HTTP Serv" and a link "[FMIDs]"; "Version" has the value "1.02.01"; "Language" has the value "English (US)"; and "Notes" is empty. At the bottom of the window, there are four buttons: "Previous", "Save", "Discard", and "Continue".

Select	Product	Description	Version	Language	Notes
<input type="checkbox"/> ▲ ○ ◆	[5655-M23]	IBM Ported Tools: HTTP Serv [FMIDs]	1.02.01	English (US)	

Previous Save Discard Continue

Figure 3-4 The IBM HTTP Server powered by Apache found

- Select the product and click **Continue**.

The Step 5 of 8 Specify order contents window opens. The following warning message is displayed:

ATTENTION: Your order contains bypassable requisites.

- You see that the message indicates that ordering z/OS and IBM Ported Tools for z/OS can be bypassed. Leave all bypassable products cleared. Click **Continue**.
- The Step 6 of 8 Select new licenses window opens. Select **System independent license** and click **Continue**.

11. The Step 7 of 8 Specify delivery options window opens. The Preferred media menu features several options. Although the quickest and simplest way to deliver the software is through the Internet, other options are available, as shown in Figure 3-5.

Shopz > My current order > Specify delivery options >

Step 7 of 8 Specify delivery options

IHS Apache - 2014-11-11 16.42.03

Specify how you would like your order delivered.

The fields indicated with an asterisk (*****) are required to complete this transaction; other fields are optional. If you do not want to provide us with the required information, please use the "Back" button on your browser to return to the previous page, or close the window or browser session that is displaying this page.

Delivery media

Preferred media*

Additional delivery options delivery media.

Internet

Select one

Internet

DVD (4.7 GB)

3590 Tape Cartridge

3592 Tape Cartridge

ge after you have selected your preferred

Figure 3-5 Selecting the delivery method

12. When the Internet option is selected, the display is updated with more information about the Shipping address, Bill to details, Payer details, Purchase Order, and Special instructions. The Purchase order section should default to Not required. Click **Continue**.
13. The Step 8 of 8 Review and submit window opens. Review the details and if all of the information is correct, click Submit.
14. A window opens in which you are prompted to confirm the order that you are about to place. Click **OK**.

15. Another a window opens that confirms that your order was placed, as shown in Figure 3-6.

Shopz > In process > Processing >

My orders

NOTE:

Order IHS Apache - 2014-11-11 16.42.03 has been submitted. You will receive an e-mail confirmation.

Create new order

Draft orders

In process

Completed

Processing

Awaiting approval

To review or process an order, click on its name. To track it, click on its status.

Load new orders into view(please be patient)

Refresh order status

In process orders

Select	Order reference number - Order name	Status
	U01535715 - IHS Apache - 2014-11-11 16.42.03 Customer number: 994616	Order Center History

Figure 3-6 Confirmation that order was placed

16. A confirmation email is sent that includes information about your order, as shown in Figure 3-7.

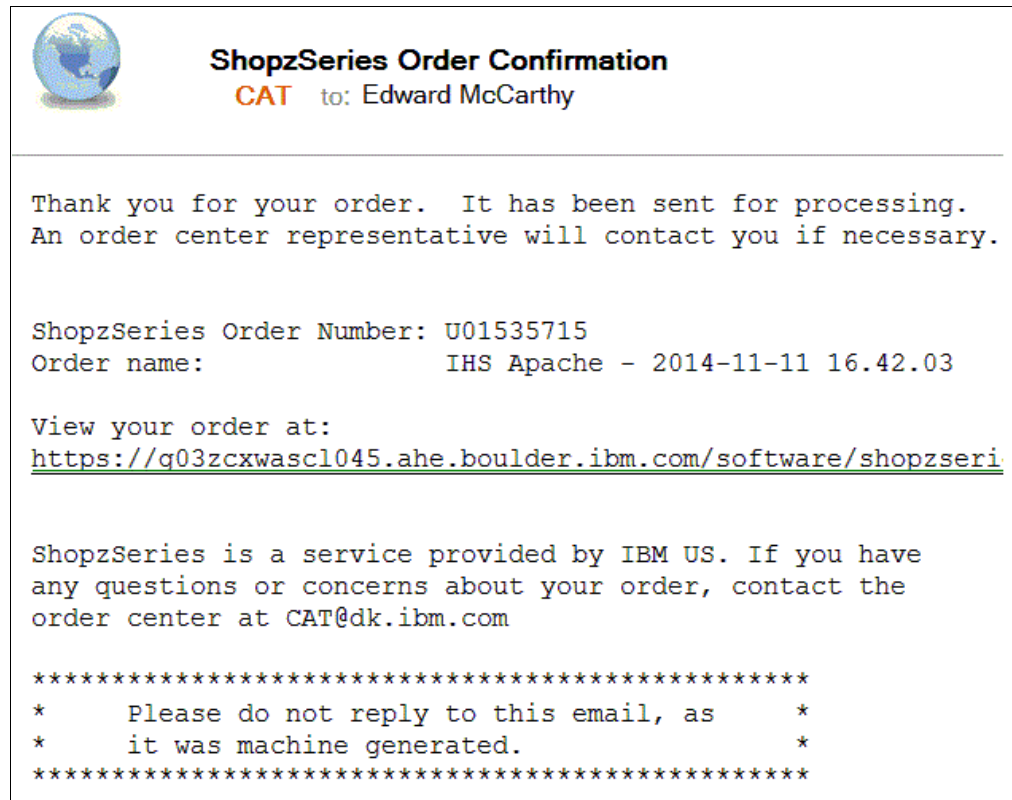


Figure 3-7 Email confirming that order is placed

3.3.3 Downloading the software

Some time later, an email is sent advising that the software is ready for download, as shown in Figure 3-8.

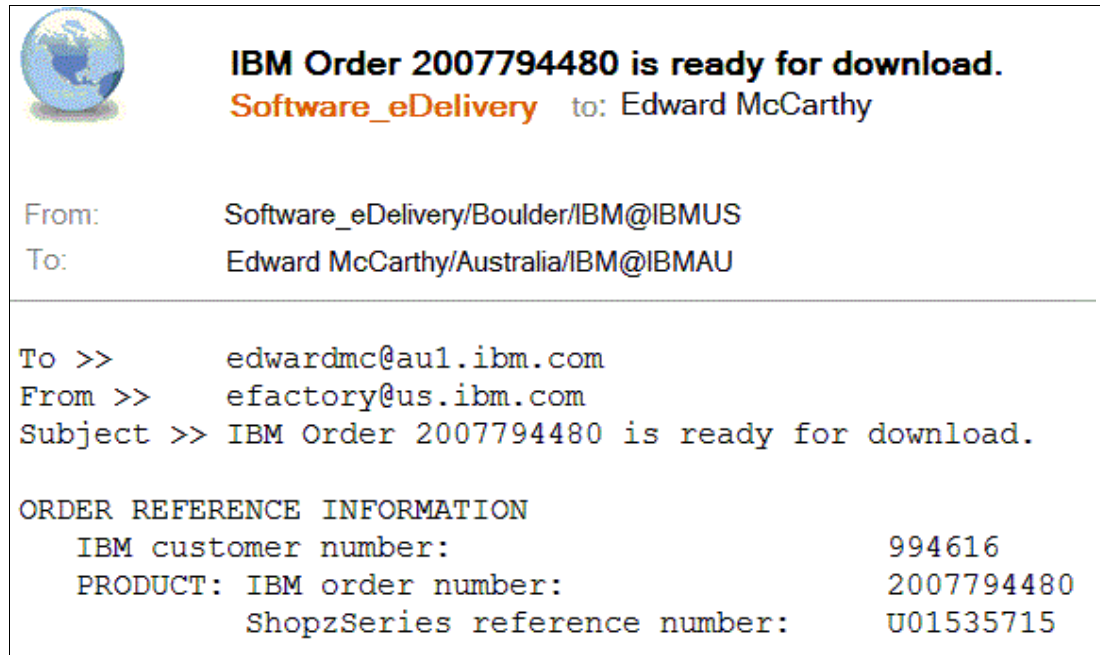


Figure 3-8 Email confirming order ready for download

Complete the following steps to download the software:

1. Log back in to Shopz and you can see that the order now showed a link to the download, as shown in Figure 3-9.

In process orders		
Select	Order reference number - Order name	Status
<input type="checkbox"/>	U01535715 - IHS Apache - 2014-11-11 16.42.03 Customer number: 994616 IBM order number: 2007794480	Physical None Internet Download

Figure 3-9 Order in Shopz with download link

2. Click link and the window that is shown in Figure 3-10 opens.

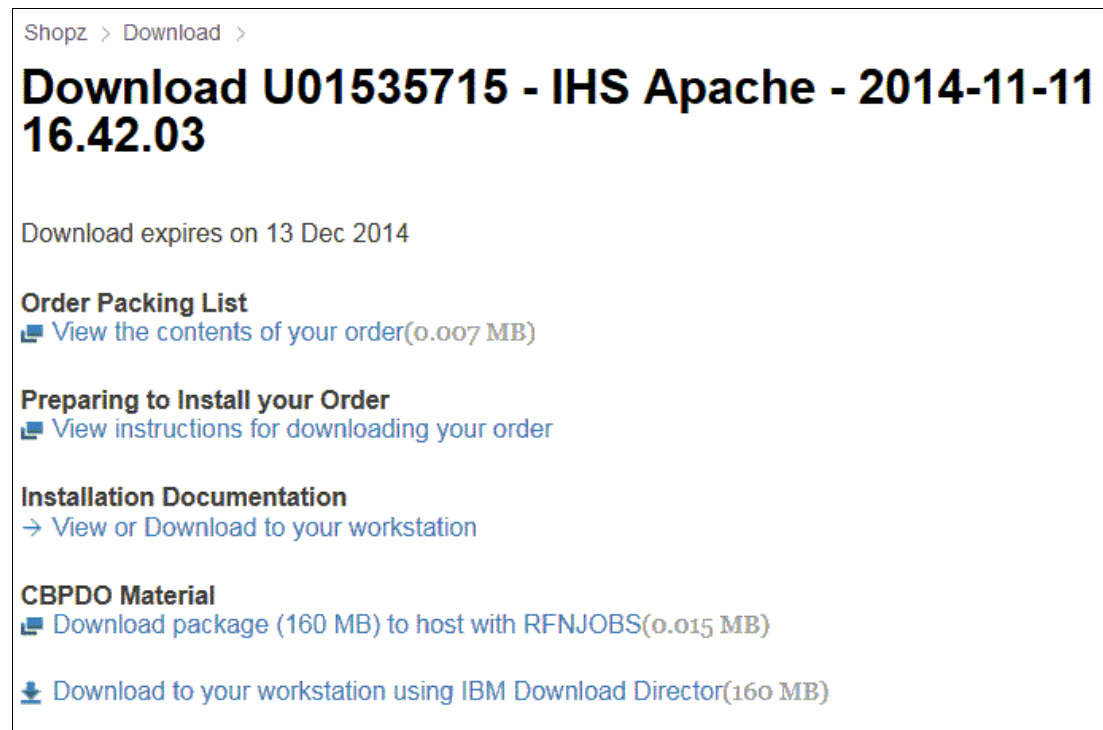


Figure 3-10 Shopz display where order can be downloaded from

3. Click **Download to your workstation using IBM Download Director**. The Download Director opens in a new browser window. When prompted, specify a new directory to which to download the product code. When the download completes, the Download Director window looks as shown in Figure 3-11.

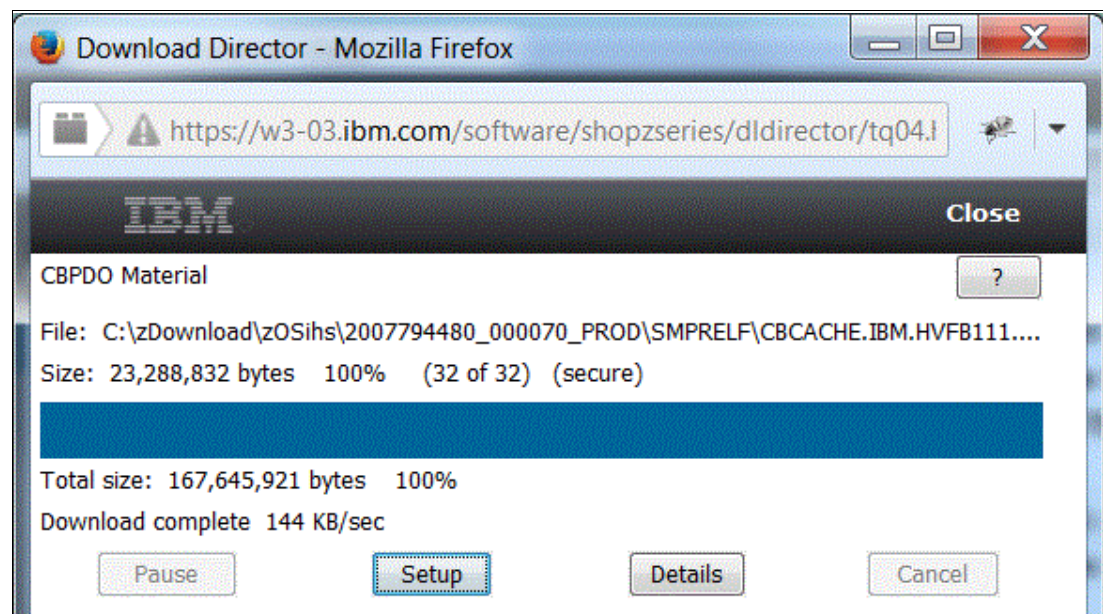


Figure 3-11 Completion of download

Note: The Download package (160 MB) to host with RFNJOBS option also can be used. This option runs a job on your z/OS LPAR that downloads the software from an IBM website direct to your z/OS LPAR. (We did not try this approach for our demonstration.) This approach requires that your z/OS LPAR can connect to the Internet. If it can connect, this approach simplifies downloading the product package onto your z/OS LPAR because you do not have to download the package to your PC and then transfer it by using FTP to your z/OS LPAR.

However, not all sites enable their z/OS LPARs to create connections to the internet, which is why we used the process that is described here.

Figure 3-12 shows the content of the directory on the PC to which the product code was downloaded.

Name	Date modified	Type	Size
SMPHOLD	13/11/2014 10:34 ...	File folder	
SMPPTFIN	13/11/2014 10:44 ...	File folder	
SMPRELF	13/11/2014 10:47 ...	File folder	
GIMPAF.XML	13/11/2014 10:34 ...	XML Docu...	20 KB
GIMPAF.XSL	13/11/2014 10:34 ...	XSL Stylesh...	5 KB
S0001.CSP.CSP.README	13/11/2014 10:34 ...	README Fi...	12 KB
S0002.CSP.STP32529.DOCLIB.pax.Z	13/11/2014 10:34 ...	Z File	32 KB
S0003.CSP.STP32529.RIMLIB.pax.Z	13/11/2014 10:34 ...	Z File	32 KB
S0005.CSP.STP32529.PGMDIR.pax.Z	13/11/2014 10:34 ...	Z File	315 KB
S0030.CSP.STP32529.GIMUNZIP	13/11/2014 10:34 ...	GIMUNZIP ...	7 KB

Figure 3-12 Contents of directory product code download into

Obtaining PTF UI22400

To resolve a packaging error that prevents successful SMP/E installation, you must order PTF UI22400 by using Shopz. After ordering this PTF from Shopz, the Download Director is used to download the PTF to the PC. This process results in a directory structure that is similar to the product code.

3.3.4 FTP product code to z/OS UNIX in z/OS

After the product code is downloaded onto your PC, it must be transferred by using FTP to a directory in the z/OS UNIX part of the z/OS LPAR.

Transferring the GIMUNZIP file first

The first job that you run is the job in the following file:

S0030.CSP.STP32529.GIMUNZIP

We recommend that you perform a separate transfer by using FTP to copy the S0030.CSP.STP32529.GIMUNZIP file to a data set on your z/OS LPAR. When we completed this transfer, we issued the commands that are shown in Example 3-1. We used the **cd IHS** command to create a second-level data set qualifier.

Example 3-1 FTP of file to data set on z/OS LPAR

```
ftp> quote site recfm=fb blksize=6160 lrecl=80
200 SITE command was accepted
ftp> quote type e
200 Representation type is Ebcidic NonPrint
ftp> cd "IHS."
250 "EDMCAR.IHS." is the working directory name prefix.
ftp> put S0030.CSP.STP32529.GIMUNZIP
200 Port request OK.
125 Storing data set EDMCAR.IHS.S0030.CSP.STP32529.GIMUNZIP
250 Transfer completed successfully.
ftp: 6399 bytes sent in 0.95Seconds 6.76Kbytes/sec.
```

STP qualifier value

In the name of the GIMUNZIP file, the third-level qualifier starts with the string STP. In our case, this third-level qualifier was STP32529. It is a different value for different orders. Make a note of this value because it is hardcoded in the supplied SMP/E Receive JCL, as described in 3.3.8, “Receiving the product code” on page 50.

You must use the value of STP32529 when a subdirectory is created to store the product code on the z/OS LPAR, as described next.

Transferring the product code

You need approximately 160 MB of disk space to store the files that make up the product code. Therefore, you must allocate a new zFS data set to store the files.

We allocated a new zFS data set that we planned to use to store the source product code and the installed product code. This approach is only an example and you can use directory names that suit your environment.

Example 3-2 shows the JCL that we ran to allocate, format, and mount a new zFS in a directory that was named /shared/ihs855.

Example 3-2 JCL to allocate, format, and mount a zFS

```
//EDMCARZ JOB (D999,MISC),'McCarthy X-8588',
//          MSGLEVEL=(1,1),
//          REGION=0M,
//          CLASS=A,
//          MSGCLASS=0
/*JOBPARM SYSAFF=SYSA
//DEFINE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE OMVS.IHS855.ZFS CLUSTER
IF LASTCC=8 THEN SET MAXCC=0
DEFINE CLUSTER (NAME(OMVS.IHS855.ZFS) -
  LINEAR MEGABYTES(600 100) SHAREOPTIONS(2))
/*
//FORMAT EXEC PGM=IOEAGFMT,REGION=0M,
```

```
// PARM=(' -aggregate OMVS.IHS855.ZFS -compat ')
//SYSPRINT DD SYSOUT=*
/*
//MOUNTZFS EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//SYSTSIN DD *
  BPXBATCH SH +
  mount -f OMVS.IHS855.ZFS +
        -o AGGRGROW +
        /shared/ihs855 ; +
  df -k | grep OMVS.IHS855.ZFS ; +
  cd /shared ; +
  chmod 775 ihs855 ; +
  ls -lrt | grep ihs855
```

We then created a subdirectory by using the commands that are shown in Example 3-3. This subdirectory into which we transfer the product code by using FTP. The name of the subdirectory we created is the third-level qualifier of the GIMUNZIP file name, as described in “STP qualifier value” on page 41.

Example 3-3 Setting up a subdirectory to store the product code

```
$ cd /shared/ihs855
$ mkdir STP32529
$ chmod 775 STP32529
$ ls -lrt
total 16
drwxrwxr-x  2 EDMCAR  SYS1          8192 Nov 13 18:58 code
```

Standard FTP does not provide a mechanism to automatically transfer all files from a current directory and all subdirectories. To transfer the contents of the subdirectories, you must change into each subdirectory to transfer by using FTP the contents of each one to the z/OS LPAR. Be sure to use binary transfer.

Example 3-4 shows the subdirectories and files in the /shared/ihs855/STP32529 directory after the FTP process is complete.

Example 3-4 Files and subdirectories on z/OS after FTP process is complete

```
$ pwd
/shared/ihs855/STP32529
$ ls -lrtR

.:
total 944
-rw-r-----  1 EDMCAR  SYS1          5040 Nov 13 19:06 GIMPAF.XSL
-rw-r-----  1 EDMCAR  SYS1        20480 Nov 13 19:06 GIMPAF.XML
-rw-r-----  1 EDMCAR  SYS1        11826 Nov 13 19:06 S0001.CSP.CSP.README
-rw-r-----  1 EDMCAR  SYS1        32256 Nov 13 19:06
S0002.CSP.STP32529.DOCLIB.pax.Z
-rw-r-----  1 EDMCAR  SYS1        32256 Nov 13 19:06
S0003.CSP.STP32529.RIMLIB.pax.Z
```

```

-rw-r----- 1 EDMCAR SYS1      6399 Nov 13 19:06 S0030.CSP.STP32529.GIMUNZIP
-rw-r----- 1 EDMCAR SYS1    322560 Nov 13 19:06
S0005.CSP.STP32529.PGMDIR.pax.Z
drwxr-x--- 2 EDMCAR SYS1      8192 Nov 13 19:06 SMPHOLD
drwxr-x--- 2 EDMCAR SYS1      8192 Nov 13 20:06 SMPPTFIN
drwxr-x--- 2 EDMCAR SYS1      8192 Nov 13 20:23 SMPRELF

```

./SMPHOLD:

total 2032

```

-rw-r----- 1 EDMCAR SYS1    1032192 Nov 13 19:07
S0004.CSP.STP32529.HOLDDATA.pax.Z

```

./SMPPTFIN:

total 206960

```

-rw-r----- 1 EDMCAR SYS1      32256 Nov 13 19:06
S0006.CBCACHE.HHAP85P.SMPMCS.pax.Z
-rw-r----- 1 EDMCAR SYS1      32256 Nov 13 19:06
S0007.CBCACHE.HOS1120.SMPMCS.pax.Z
-rw-r----- 1 EDMCAR SYS1      32256 Nov 13 19:07
S0008.CBCACHE.HVFB111.SMPMCS.pax.Z
-rw-r----- 1 EDMCAR SYS1    18115072 Nov 13 19:25
S0009.CBCACHE.HHAP85P.PTF.UI17041.pax.Z
-rw-r----- 1 EDMCAR SYS1    1397760 Nov 13 19:29
S0011.CBCACHE.HOS1120.PTF.UA57163.pax.Z
-rw-r----- 1 EDMCAR SYS1    23586304 Nov 13 19:39
S0010.CBCACHE.HHAP85P.PTF.UI20159.pax.Z
-rw-r----- 1 EDMCAR SYS1    18388480 Nov 13 19:45
S0012.CBCACHE.HOS1120.PTF.UA63842.pax.Z
-rw-r----- 1 EDMCAR SYS1    2717696 Nov 13 19:47
S0013.CBCACHE.HOS1120.PTF.UA67935.pax.Z
-rw-r----- 1 EDMCAR SYS1    797184 Nov 13 19:47
S0014.CBCACHE.HOS1120.PTF.UA67952.pax.Z
-rw-r----- 1 EDMCAR SYS1    2870784 Nov 13 19:49
S0016.CBCACHE.HOS1120.PTF.UA70330.pax.Z
-rw-r----- 1 EDMCAR SYS1    2515968 Nov 13 19:51
S0017.CBCACHE.HOS1120.PTF.UA71137.pax.Z
-rw-r----- 1 EDMCAR SYS1    2870784 Nov 13 19:53
S0018.CBCACHE.HOS1120.PTF.UA71772.pax.Z
-rw-r----- 1 EDMCAR SYS1    3685376 Nov 13 19:58
S0019.CBCACHE.HOS1120.PTF.UA72557.pax.Z
-rw-r----- 1 EDMCAR SYS1    3299328 Nov 13 20:04
S0020.CBCACHE.HOS1120.PTF.UA73914.pax.Z
-rw-r----- 1 EDMCAR SYS1    1290240 Nov 13 20:06
S0021.CSP.STP32529.ASSIGNS.pax.Z
-rw-r----- 1 EDMCAR SYS1      32256 Nov 13 20:06
S0022.CSP.STP32529.PRODDATA.pax.Z
-rw-r----- 1 EDMCAR SYS1    24071168 Nov 13 20:15
S0015.CBCACHE.HOS1120.PTF.UA68137.pax.Z

```

./SMPRELF:

total 123376

```

-rw-r----- 1 EDMCAR SYS1      32256 Nov 13 20:06 CBCACHE.IBM.HHAP85P.F1.pax.Z
-rw-r----- 1 EDMCAR SYS1      32256 Nov 13 20:06 CBCACHE.IBM.HHAP85P.F2.pax.Z
-rw-r----- 1 EDMCAR SYS1      32256 Nov 13 20:15 CBCACHE.IBM.HOS1120.F1.pax.Z

```

```

-rw-r----- 1 EDMCAR SYS1      21385728 Nov 13 20:22
CBCACHE.IBM.HHAP85P.F3.pax.Z
-rw-r----- 1 EDMCAR SYS1      32256 Nov 13 20:23 CBCACHE.IBM.HVFB111.F1.pax.Z
-rw-r----- 1 EDMCAR SYS1     18284544 Nov 13 20:41
CBCACHE.IBM.HOS1120.F2.pax.Z
-rw-r----- 1 EDMCAR SYS1     23288832 Nov 13 20:42
CBCACHE.IBM.HVFB111.F2.pax.Z

```

FTP UI22400 to z/OS

We used a similar process to binary transfer the contents of the UI22400 PTF to our z/OS LPAR. Example 3-5 shows the contents of the directory on z/OS after this transfer was complete.

Example 3-5 Contents of directory that includes the contents of PTF UI22400

```

$ pwd
/shared/ihs855/UI22400
$ ls -lrtR

.:
total 64
drwxr-xr-x  2 EDMCAR SYS1      8192 Nov 17 17:11 SMPPTFIN
drwxr-xr-x  2 EDMCAR SYS1      8192 Nov 17 17:11 SMPHOLD
-rw-r----- 1 EDMCAR SYS1     4800 Nov 17 17:11 GIMPAF.XSL
-rw-r----- 1 EDMCAR SYS1     2720 Nov 17 17:11 GIMPAF.XML

./SMPPTFIN:
total 35280
-rw-r----- 1 EDMCAR SYS1    18031104 Nov 17 17:11
S0001.SHOPZ.S2666231.SMPMCS.pax.Z

./SMPHOLD:
total 2096
-rw-r----- 1 EDMCAR SYS1    1064448 Nov 17 17:11
S0002.SHOPZ.S2666231.SMPHOLD.pax.Z

```

3.3.5 First job to run: GIMUNZIP

We performed the FTP file transfer that is described in “Transferring the GIMUNZIP file first” on page 40 to add it to the data set. Then, we ran the following JCL job GIMUNZIP:

EDMCAR.IHS.S0030.CSP.STP32529.GIMUNZIP.

We modified EDMCAR.IHS.S0030.CSP.STP32529.GIMUNZIP (as shown in Example 3-6) to reflect our environment.

Example 3-6 Modified GIMUNZIP JCL

```

//UNZIP   EXEC PGM=GIMUNZIP,REGION=OM,PARM='HASH=NO'    <=== NOTE 1
//SYSUT3  DD UNIT=SYSALLDA,SPACE=(CYL,(50,10))
//SYSUT4  DD UNIT=SYSALLDA,SPACE=(CYL,(25,5))
//SMPJHOME DD PATH='/usr/lpp/java/J6.0.1_64/'          <===NOTE 2
//SMPCPATH DD PATH='/usr/lpp/smp/classes/'              <===NOTE 2
//SMPOUT  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*

```

```
//SMPDIR DD PATH='/shared/ihs855/STP32529', <=== NOTE 3
//          PATHDISP=KEEP
//SYSIN DD *
<GIMUNZIP>
<ARCHDEF
name="S0002.CSP.STP32529.DOCLIB.pax.Z"
volume="PTS002"
newname="EDMCAR.IHS.DOCLIB">
</ARCHDEF>
<ARCHDEF
name="S0003.CSP.STP32529.RIMLIB.pax.Z"
volume="PTS002"
newname="EDMCAR.IHS.RIMLIB">
</ARCHDEF>
<ARCHDEF
name="S0005.CSP.STP32529.PGMDIR.pax.Z"
volume="PTS002"
newname="EDMCAR.IHS.PGMDIR">
</ARCHDEF>
</GIMUNZIP>
```

The JCL that is shown in Example 3-6 refers to the following files:

- ▶ S0002.CSP.STP32529.DOCLIB.pax.Z
- ▶ S0003.CSP.STP32529.RIMLIB.pax.Z
- ▶ S0005.CSP.STP32529.PGMDIR.pax.Z

The GIMUNZIP job looks for these three files in the directory that is specified by the SMPDIR DD card, which in our case was /shared/ihs855/STP32529. The files were stored in that directory when we transferred the product code, as described in “Transferring the product code” on page 41.

We ran the job. After the job was successfully completed, we now had the following three data sets:

- ▶ EDMCAR.IHS.DOCLIB
- ▶ EDMCAR.IHS.PGMDIR
- ▶ EDMCAR.IHS.RIMLIB

3.3.6 Second job to run: UNZIPJCL

The second job to be run is the member that is named UNZIPJCL in the EDMCAR.IHS.RIMLIB data set. When this job is run, it creates a data set, which in our case we named EDMCAR.IHS.SAMPLE.JOBS.

Example 3-7 shows the JCL for the UNZIPJCL job after it was modified.

Example 3-7 Modified UNZIPJCL job

```
//UNZIP EXEC PGM=GIMUNZIP,REGION=OM,PARM='HASH=NO'
//SYSUT3 DD UNIT=SYSALLDA,SPACE=(CYL,(10,10))
//SYSUT4 DD UNIT=SYSALLDA,SPACE=(CYL,(15,5))
//SMPJHOME DD PATH='/usr/lpp/java/J6.0.1_64/'
//SMPCPATH DD PATH='/usr/lpp/smp/classes/'
//SMPOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SMPDIR DD PATH='/shared/ihs855/STP32529/SMPRELF/',
```

```
//          PATHDISP=KEEP
//SYSIN    DD *
<GIMUNZIP>
<ARCHDEF
name="CBCACHE.IBM.HHAP85P.F1.pax.Z"
volume="PTS002"
newname="edmcар.іhs.sample.jobs">
</ARCHDEF>
</GIMUNZIP>
/*
```

We ran this job and it created the EDMCAR.IHS.SAMPLE.JOBS data set. This data set contains sample SMP/E jobs. In the following sections, we describe how we modified and ran these jobs to complete the SMP/E process.

3.3.7 Setting up SMP/E

For the purposes of describing for this document the SMP/E installation process, we did not want to use the SMP/E environment on the z/OS LPAR. Instead, we created a SMP/E environment to be used only for installing IBM HTTP Server powered by Apache.

We used the sample JCL in SYS1.SAMPLIB(GIMSAMPU) to create this environment. We copied this JCL and modified it for our use. An important change was to allocate a larger SMPPTS, as shown in Example 3-8.

Example 3-8 Allocating a larger SMPPTS

```
//SMPPTS   DD DSN=EDMCAR.IHS855.SMPPTS,
//          DISP=(NEW,CATLG),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=0,DSORG=PO),
//          DSNTYPE=LIBRARY,
//          UNIT=SYSDA,
//          SPACE=(CYL,(550,50,50))
```

On your system, you can use your SMP/E environment.

If you have an older version of IBM HTTP Server powered by Apache installed, there are JCL samples in EDMCAR.IHS.SAMPLE.JOBS to assist with removing the older version.

Allocating target and distribution data sets

The supplied JCL in EDMCAR.IHS.SAMPLE.JOBS(HAPALL02) allocates the SMP/E target and distribution data sets for the IBM HTTP Server powered by Apache product.

We modified the supplied JCL in SAMPLE.JOBS(HAPALL02), as shown in Example 3-9.

Example 3-9 Modified HAPALL02 JCL

```
//ALLOCT EXEC ALLOCTGT,
//          HLQ=EDMCAR.HAP.V855, * HAP is the default
//          DSP=CATLG,      * CATLG is the default
//          TVOL1=PTS006, * No default; volume1 for target library
//          TVOL2=PTS007 * No default; volume2 for target library
//*
//ALLOCD EXEC ALLOCDLB,
//          HLQ=EDMCAR.HAP.V855, * HAP is the default
```



```

//          DSP=CATLG,      * CATLG is the default
//          DVOL=PTS006    * No default; volume for dist. libraries
//
// *
// * Note the // above to stop following step from running
// *
// * Did not use the following JCL as require zFS not HFS
// *
// *****
// *   The following step executes the PROC to allocate a new      *
// *   HFS data set for IBM HTTP Server V8.5                      *
// *****
// *
//ALLOCH EXEC ALLOCHFS,
//          HLQ1=HAP.V8R5 * HAP.V8R5 is the default
//          DSP=CATLG,      * CATLG is the default
//          HVOL=hhhhh1    * No default; volume for HFS data set

```

When we ran this JCL, we noticed that it allocated an HFS rather than a zFS. Because the use of HFS is not recommended, we added a line in the JCL to stop the HFS from being allocated by this JCL. We then set up the JCL as shown in Example 3-10 to allocate and format a zFS. In the last step, we mounted the zFS at the directory where we want the product code to be stored. In our case, the target directory was /shared/IHSA/V8R5M5.

Example 3-10 JCL to allocate a zFS to store the product code

```

//DEFINE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE OMVS.IHS.V8R5M5.ZFS CLUSTER
IF LASTCC=8 THEN SET MAXCC=0
DEFINE CLUSTER (NAME(OMVS.IHS.V8R5M5.ZFS) -
  LINEAR MEGABYTES(600 100) SHAREOPTIONS(2))
/*
//FORMAT EXEC PGM=IOEAGFMT,REGION=0M,
// PARM=(' -aggregate OMVS.IHS.V8R5M5.ZFS -compat ')
//SYSPRINT DD SYSOUT=*
/*
//MOUNTZFS EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//SYSTSIN DD *
BPXBATCH SH +
cd /shared/IHSA ; +
mkdir V8R5M5 ; +
mount -f OMVS.IHS.V8R5M5.ZFS +
-o AGGRGROW +
  /shared/IHSA/V8R5M5 ; +
df -k | grep OMVS.IHS.V8R5M5.ZFS ; +
chmod 775 V8R5M5 ; +
cd /shared/IHSA/V8R5M5 ; +
mkdir IBM ; +
chmod 775 IBM ; +

```

```
pwd; +  
ls -Rlrt
```

It is good practice to have the directory that was used for the SMP/E process separate from the directory that is used in your runtime environment. In this case, we are installing the IBM HTTP Server powered by Apache product code into the directory at /shared/IHSA/V8R5M5.

After we completed the SMP/E process, we created a zFS at a directory that was named /usr/lpp/zWebSphere_0M/V8R5M5, and used DFDSS to copy the zFS at /shared/IHSA/V8R5M5 to /usr/lpp/zWebSphere_0M/V8R5M5.

The zFSs at these two locations are mounted as Read Only so that they are not accidentally updated when they are not updated through SMP/E.

This process also allocates a subdirectory that is named IBM that is required for the SMP/E process.

Adding DDDEFs to SMP/E

The supplied JCL in EDMCAR.IHS.SAMPLE.JOBS(HAPDDDE2) defines the DDDEFs in the SMP/E environment for the IBM HTTP Server powered by Apache product.

We modified the JCL in EDMCAR.IHS.SAMPLE.JOBS(HAPDDDE2). The first part of Example 3-11 shows how the comments in the JCL reflect the changes we made, whereas the bottom part shows how we changed the JCL to reflect the target directory we are using.

Example 3-11 Modified HAPDDDE2 JCL

- 2) Change EDMCAR.IHS855.GLOBAL.CSI to the data set name of your global data set
- 3) Change TARGET to the name of your target zone
- 4) Change DLIB to the name of your distribution zone
- 5) Change EDMCAR.HAP.V855 to the appropriate high-level qualifier
- 6) This job uses the recommended data set placement for the target libraries:
Change PTS006 to the volser for first volume for the target libraries (TVOL1).
Change PTS007 to the volser for second volume for the target libraries (TVOL2).
- 7) Change PTS006 to the volser of the distribution volume.

```
//DEFPATH EXEC PGM=GIMSMP,REGION=4096K  
//SMPCSI DD DSN=EDMCAR.IHS855.GLOBAL.CSI,  
// DISP=SHR  
//SMPCNTL DD *  
SET BDY(TARGET) . /* CHANGE -PathPrefix- */  
ZONEEDIT DDDEF.  
CHANGE PATH('/usr/lpp/IHSA/V8R5/'*,  
            '/shared/IHSA/V8R5M5/'*).  
ENDZONEEDIT.
```

Increasing DSSPACE

Check the disk space settings for DSSPACE in the global zone of your SMP/E environment. If they are the original defaults, they are too small and the Receive fails. Example 3-12 shows the values that we used that allowed for a successful Receive.

Example 3-12 DSSPACE setting in the SMP/E Global zone

```
OPTIONS ENTRY GOPT - DSSPACE/DSPREFIX
====>
```

The OPTIONS entry contains the DSSPACE (data set space allocation) values and the DSPREFIX for the SMPTLIB Data sets.

Verify or enter the values for OPTIONS entry GOPT:

```
DSPREFIX      ===> EDMCAR.IHS855.SMPTLIB
                (Data set name prefix, up to
                26 characters)
PRIMARY TRACKS ===> 2000      (Up to 4 numeric
                                characters)
SECONDARY TRACKS ===> 200      (Up to 4 numeric
                                characters)
DIRECTORY BLOCKS ===> 1000     (Up to 4 numeric
                                characters)
```

Adjusting SYSUT4 space values

Check the disk space settings for DDDEF SYSUT4 in the global zone of your SMP/E environment. If they are the original defaults, they are too small and the Receive fails. Example 3-13 shows the values that we used that allowed for a successful Receive.

Example 3-13 SYSUT4 settings in the SMP/E Global zone

```
DDDEF ENTRY SYSUT4 - LIBRARY TYPE
====>
```

Enter Library DDDEF data to allocate DD statements for data sets to be dynamically allocated during SMP/E processing. Values must conform to JCL conventions. However, no parenthesis can be entered.

```
DATA SET NAME ===>
                (data set name, maximum 44 characters)
INITIAL DISP  ===>          (OLD,SHR,MOD,NEW)
FINAL DISP   ===>          (KEEP,DELETE,CATALOG)
UNIT         ===> SYSALLDA (unit type if not cataloged)
VOLUME       ===>          (volume serial)
SPACE UNITS  ===> CYL      (TRK, CYL, or block length)
PRIMARY      ===> 500      (primary space)
SECONDARY    ===> 100      (secondary space)
DIR          ===>          (Number of directory blocks)
SYSOUT       ===>          (SYSOUT class)
WAITFORDSN   ===> NO       (YES or NO)
PROTECT      ===> NO       (YES or NO)
SMS OPTIONS  ===> NO       (YES or NO to edit SMS Options)
```

Press ENTER to save the changes.

3.3.8 Receiving the product code

The supplied RIMLIB(RCVPDO) JCL is used to receive the product code into the SMP/E environment. We modified it as shown in Example 3-14. We added a RECEIVE control statement to receive the UI22400 PTF.

Example 3-14 Modified RCVPDO JCL

```
//SMPER1 EXEC PGM=GIMSMP,REGION=OM,
//          PARM='PROCESS=WAIT',
//          DYNAMNBR=120
//SMPCSI DD DISP=SHR,DSN=EDMCAR.IHS855.GLOBAL.CSI <=== NOTE 1
//SMPNTS DD PATHDISP=KEEP,
//          PATH='/shared/ihs855/' <=== NOTE 2
//SMPOUT DD SYSOUT=*
//SMPRPT DD SYSOUT=*
//SMPLIST DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//*****
/* AS SHIPPED BY IBM, RCVPDO IS SET UP TO RECEIVE
/* THE FMIDS, PTFS AND HOLDDATA
//*****
//SMPCNTL DD *
SET BOUNDARY (GLOBAL) .
RECEIVE
FROMNTS(STP32529)
.
RECEIVE
FROMNTS(UI22400)
.
```

3.3.9 Applying the product code

The supplied EDMCAR.IHS.SAMPLE.JOBS(HAPAPPL2) JCL is used to apply the product code into the SMP/E environment. We modified it as shown in Example 3-15. We added the UI22400 PTF to the APPLY statement.

Example 3-15 Modified SMP/E Apply JCL

```
//STEP1 EXEC PGM=GIMSMP,REGION=OM,TIME=NOLIMIT
//SMPCSI DD DSN=EDMCAR.IHS855.GLOBAL.CSI,DISP=SHR
//SMPCNTL DD *
SET BOUNDARY(TARGET) .
APPLY
FORFMID(HHAP85P)
SELECT(HHAP85P,UI22400)
GROUPEXTEND(NOAPARS,NOUSERMODS)
BYPASS(HOLDSYS,XZIFREQ).
/*
```

We ran an APPLY CHECK first. The key point to check is the directory the product code is stored. You perform this check by reviewing the value that is specified for SHAPBIN2 in the File Allocation Report, as shown in Example 3-16 on page 51.

Example 3-16 SMP/E report showing target directory

SMP APPLY CHECK FILE ALLOCATION REPORT

DDNAME	DDDEFNAM	SMPDDNAM	TYPE	-----DATA SET OR PATH-----
SHAPBIN2	SHAPBIN2		PATH	'/shared/IHSA/V8R5M5/IBM/'

We then removed the CHECK keyword and ran the job again, which completed successfully.

We checked our target directory and saw the content that is shown in Example 3-17.

Example 3-17 Content of installed IBM HTTP Server powered by Apache

Directory List

Select one or more files with / or action codes. If / is used also action from the action bar otherwise your default action will be us with S to use your default action. Cursor select can also be used navigation. See help for details.

```
EUID=354457    /shared/IHSA/V8R5M5/
  Type  Permission  Changed-EST5EDT  Owner    Filename      R
_ Dir    rwxrwxr-x   2014-11-17 18:51  EDMCAR    .
_ Dir    rwxr-xr-x    2014-11-14 02:01  HUTCH     ..
_ Dir    rwxrwxrwx   2014-09-05 08:24  QWER01    .3lbit
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    bin
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    build
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    conf
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    error
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    example_module
_ File   rwxr-xr-x   2014-11-17 18:51  QWER01    HAPBB001.zip
_ File   rwxr-xr-x   2014-11-17 18:51  QWER01    HAPBE001.sh
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    htdocs
_ Dir    rwxr-xr-x   2014-11-17 18:51  EDMCAR    IBM
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    icons
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    include
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    lib
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    man
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    modules
_ File   rwxr-xr-x   2014-05-15 18:59  QWER01    notices
_ Dir    rwxr-xr-x   2014-09-05 08:24  QWER01    readme
_ File   rwxr-xr-x   2014-11-17 18:51  QWER01    readme.txt
_ File   rwxr-xr-x   2014-09-05 08:24  QWER01    version.signature
```

Example 3-17 indicates that the product code for IBM HTTP Server powered by Apache was installed successfully into the target directory in the z/OS UNIX environment on z/OS.

Browsing the version.signature file showed that it contained IBM HTTP Server 8.5.5.3, which shows the version of the IBM HTTP Server powered by Apache that was installed.

3.3.10 Accepting the product code

The supplied EDMCAR.IHS.SAMPLE.JOBS(HAPACCE2) JCL is used to accept the product code into the SMP/E environment. We modified it as shown in Example 3-18. We added the UI22400 PTF to the ACCEPT statement.

Example 3-18 Modified Accept SMP/E JCL

```
//STEP1 EXEC PGM=GIMSMP,REGION=0M,TIME=NOLIMIT
//SMPCSI DD DSN=EDMCAR.IHS855.GLOBAL.CSI,DISP=SHR
//SMPCNTL DD *
    SET BOUNDARY(DLIB) .
    ACCEPT CHECK
    FORFMID(HHAP85P)
    SELECT(HHAP85P,UI22400)
    GROUPEXTEND(NOAPARS,NOUSERMODS)
    BYPASS(HOLDSYS) .
```

We ran an ACCEPT CHECK, which completed successfully. Then, we removed the CHECK keyword and ran the actual accept, which also completed successfully.

3.3.11 Summary

We now completed the SMP/E installation of the IBM HTTP Server powered by Apache product. The product code is installed and ready for use in the target directory in the z/OS UNIX environment on the z/OS LPAR.

We can now proceed with setting up a server, as described in IBM HTTP 3.5, “Sample real-world setup process” on page 53.

3.4 Installation when a component of another IBM product

IBM HTTP Server powered by Apache can be delivered as part of other IBM products, such as WebSphere Application Server for z/OS. The installation of products, such as WebSphere Application Server for z/OS, requires the use of IBM Installation Manager. This section provides resources that you can use to find more information about how you use IBM Installation Manager to install the IBM HTTP Server powered by Apache.

When IBM HTTP Server powered by Apache is supplied with other IBM products, such as WebSphere Application Server, a repository file that contains the IBM HTTP Server powered by Apache product code also is supplied. You use IBM Installation Manager to install IBM HTTP Server powered by Apache from this repository file.

For more information about how to use the Installation Manager on z/OS, see the IBM Techdoc that is available at the following website. Although the information is not specifically for IBM HTTP Server powered by Apache, the process is the same:

<https://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102014>

The IBM Techdoc describes the following steps:

1. Creating the Installation Manager by using the Installation Manager toolkit.
2. Creating the WebSphere on z/OS V8 file system components on a /Service mount point.
3. Using the Installation Manager to populate the file system.

At the end of step 3, the product code is available in a zFS mounted at a directory in the z/OS UNIX environment.

For more information about obtaining IBM Installation Manager, see this WebSphere Application Server V8.5 IBM Knowledge Center website:

<https://ibm.biz/Bdrt44>

For more information about installing IBM Installation Manager on z/OS, see this WebSphere Application Server V8.5 IBM Knowledge Center website:

<https://ibm.biz/Bdrt4s>

For more information about how to use Installation Manager to complete the installation of IBM HTTP Server powered by Apache on z/OS, see this WebSphere Application Server website:

<https://ibm.biz/Bdrt4j>

For more information about how entitled customers can install WebSphere Application Server for z/OS components (such as IBM HTTP Server powered by Apache) directly without the use of SMP/E, see this website:

<https://www.ibm.com/support/docview.wss?uid=swg21659636>

On our z/OS LPAR, the product code for IBM HTTP Server powered by Apache was at `/ihs/usr/lpp/IHSA/V8R5` and an updated version was provided at `/usr/lpp/zWebSphereEM1_IHSA /V8R5`.

3.5 Sample real-world setup process

This section describes the steps that we used to set up an IBM HTTP Server as we do at a client site.

The steps assume that the file system that contains the IBM HTTP Server powered by Apache product code is mounted at `/ihs/usr/lpp/IHSA/V8R5`.

For IBM HTTP Server powered by Apache V9 in z/OS 2.2, we completed the process that is described in this section to set up a server to verify that the process is unchanged.

3.5.1 Defining a configuration directory

We created a new directory in the root that is named `ihsconfig`. This directory stores configurations for new IBM HTTP Servers powered by Apache.

We created a new zFS and mounted it read/write at the `ihsconfig` directory. This configuration avoids the possibility of filling up the zFS that backs the root directory.

We then created the following subdirectories:

- ▶ `ihs`: The directory under which is stored the configurations for new IBM HTTP Servers powered by Apache.
- ▶ `home`: The directory to use for the home directory for user IDs that are associated with running IBM HTTP Servers powered by Apache.

Directories for first IBM HTTP Server powered by Apache

Our first IBM HTTP Server powered by Apache is called IHSAE001. We created the following directories:

- ▶ /ihsconfig/ihs/ihsae001
- ▶ /ihsconfig/home/ihsae001
- ▶ /ihsconfig/home/ihsaestc

You can use any directory as the home location for your IBM HTTP Server. A directory, such as /ihsconfig/ihs/ihsae001, is used to reinforce the concept that the use of someone's home directory (such as /u/edward) is not a best practice.

3.5.2 Defining a user ID

We created a group by using the following command:

```
addgroup ihsrb13 omvs(gid(36000))
```

We then created a user ID that is named IHSAE001, which is used to create a server instance of an IBM HTTP Server powered by Apache. This user ID must have an OMVS segment and a home directory in the z/OS UNIX environment. This user ID is not used as the user ID to run the started task that runs the IBM HTTP Server powered by Apache. For more information about creating that user ID, see 3.5.3, “Defining a protected user ID for the started task” on page 55.

Note: There is no requirement for this user ID to have an OMVS UID of zero. It is recommended that the user does not have an OMVS UID of zero.

Example 3-19 shows the RACF commands that we used to define the user ID.

Example 3-19 Defining user ID for the server

```
adduser ihsae001 df1tgrp(ihsrb13) name('IHS Server 1 Edward') omvs(uid(35001)
home('/ihsconfig/home/ihsae001') program('/bin/sh'))

alu ihsae001 password(ihsrb13) noexpire
```

The output from issuing the **LU IHSAE001 OMVS** command in Example 3-20 shows how we set up this user ID.

Example 3-20 Listing of IHSAE001 user ID

```
USER=IHSAE001  NAME=IHS SERVER 1 EDWARD  OWNER=EDMCAR  CREATED=13.164
DEFAULT-GROUP=IHSRB13  PASSDATE=13.164  PASS-INTERVAL=180  PHRASEDATE=N/A
ATTRIBUTES=NONE
REVOKE DATE=NONE  RESUME DATE=NONE
LAST-ACCESS=13.177/07:22:50
CLASS AUTHORIZATIONS=NONE
NO-INSTALLATION-DATA
NO-MODEL-NAME
LOGON ALLOWED  (DAYS)  (TIME)
-----
ANYDAY
GROUP=IHSRB13  AUTH=USE  CONNECT-OWNER=EDMCAR  CONNECT-DATE=13.164
CONNECTS= 178  UACC=NONE  LAST-CONNECT=13.177/07:22:50
CONNECT ATTRIBUTES=NONE
```



```
REVOKE DATE=NONE RESUME DATE=NONE
SECURITY-LEVEL=NONE SPECIFIED
CATEGORY-AUTHORIZATION
  NONE SPECIFIED
SECURITY-LABEL=NONE SPECIFIED
```

OMVS INFORMATION

```
UID= 0000035001
HOME= /ihsconfig/home/ihsae001
PROGRAM= /bin/sh
CPUTIMEMAX= NONE
ASSIZEMAX= NONE
FILEPROCMAx= NONE
PROCUSERMAX= NONE
THREADSMAX= NONE
MMAPAREAMAX= NONE
```

Setting owner and permissions

We then issued the following commands to set the owner and group of the directories:

```
cd /ihsconfig
EDMCAR:/ihsconfig: >ls -lrt
total 32
drwxr-xr-x  3 EDMCAR  SYS1          8192 Oct 15 21:24 ihs
drwxr-xr-x  3 EDMCAR  SYS1          8192 Oct 15 21:24 home
EDMCAR:/ihsconfig: >chown ihsae001:ihsrb13 ihs home
EDMCAR:/ihsconfig: >ls -lrt
total 32
drwxr-xr-x  3 IHSAE001 IHSRB13      8192 Oct 15 21:24 ihs
drwxr-xr-x  3 IHSAE001 IHSRB13      8192 Oct 15 21:24 home
```

3.5.3 Defining a protected user ID for the started task

It is a best practice to run the started task that runs the IBM HTTP Server powered by Apache under a protected user ID. A protected user ID is one that includes the NOPASSWORD attribute, which means that the user ID cannot be used to log on to z/OS.

Example 3-21 shows the command we used to define this user ID.

Example 3-21 RACF command used to define protected user ID

```
adduser ihsaestc dfltgrp(ihsrb13) name('Apache Started Task ID') nopassword
omvs(uid(35065) home('/ihsconfig/home/ihsaestc') program('/bin/sh'))
```

You see the following output from an LU **ihsaestc** command:

```
ATTRIBUTES=PROTECTED
```

This user ID was used when certificates and a key ring for SSL support was created, as described in 7.4.2, “Creating required certificates” on page 123.

3.5.4 Creating the IHS

Before issuing the command to create the server instance, you should review the umask setting.

The typical default value is 0022, which results in the directories and files that are created having permissions of 755, which means any user can view the files.

You might want to consider setting the umask to 0007 by using the **umask 007** command.

The files and directories that are created all feature public permissions set to 0, which means they cannot be accessed by anyone other than the owner or someone who is a member of the group.

For more information about setting up an IBM HTTP Server powered by Apache, see this website:

<https://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101170>

Using the advice in that Techdoc, we then issued the commands shown in Example 3-22.

Example 3-22 Creating the configuration for IBM HTTP Server powered by Apache

```
su - ihsae001
cd /ihs/usr/lpp/IHSA/V8R5/bin
./install_ihs /ihsconfig/ihs/ihsae001 8230
```

Before the command that is shown in Example 3-22 is issued, ensure that you created the target directory from the ihsae001 user ID, as described in “Directories for first IBM HTTP Server powered by Apache” on page 54. Also, you must set the owner and permissions, as described in “Setting owner and permissions” on page 55.

The last value of 8230 in the command that is shown in Example 3-22 is the TCP/IP port on which the created server listens when started. The command updates the Listen directive in httpd.conf to the value you specify.

The output from issuing the commands that are shown in Example 3-22 is shown in Example 3-23.

Example 3-23 Output from creating the configuration

```
Copying install directory and creating symlinks...
Updating install paths...
cmd: /ihs/usr/lpp/IHSA/V8R5/bin/postinst -i /ihsconfig/ihs/ihsae001 -t install -v
PORT=8230 -v SERVERNAME=wtsc55.itso.ibm.com
Updating permissions for WebSphere Application Server admin console...
```

3.5.5 Defining a RACF STARTED rule

We issued the RACF commands that are shown in Example 3-24 to define the user ID for the started task that we set up to run the server under.

Example 3-24 RACF rules to map started task to a user ID

```
RDEFINE STARTED IHSAE001.* STDATA(USER(IHSAESTC))
SETOPTS RACLIST(STARTED) REFRESH
```

3.5.6 Creating a Started Task to run the IHS

We created a started task catalog procedure that is named IHSAE001 in SYS1.PROCLIB, as shown in Example 3-25.

Example 3-25 Started task JCL

```
/*-----  
//IHSAE001 PROC ACTION='start',  
//          DIR='/ihsconfig/ihs/ihsae001',  
//          CONF='conf/httpd.conf'  
/*-----  
//IHS      EXEC PGM=BPXBATCH,  
// PARM='SH &DIR/bin/apachectl -k &ACTION -f &CONF -DNO_DETACH',  
// MEMLIMIT=512M  
//STDOUT   DD PATH='&DIR/logs/proc.output',  
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),  
//          PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)  
//STDERR   DD PATH='&DIR/logs/proc.errors',  
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),  
//          PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)  
//          PEND
```

We then issued the **S IHSAE001** command to start this started task. Unlike IBM HTTP Server powered by Domino, which has only one started task running when it started, you see that several started tasks started.

Example 3-26 shows the started tasks that are present after we started our IHSAE001 server.

Example 3-26 Started tasks running after starting IHSAE001

```
SDSF DA SC55      SC55      PAG 0 CPU/L/Z  3/ 2  
COMMAND INPUT ==>  
NP  JOBNAME  StepName ProcStep JobID   Owner  
    IHSAE001 STEP1      STC18072 IHSAESTC  
    IHSAE001 STEP1      STC18068 IHSAESTC  
    IHSAE001 IHSAE001 *OMVSEX STC18082 IHSAESTC  
    IHSAE001 STEP1      STC18086 IHSAESTC  
    IHSAE001 STEP1      STC18094 IHSAESTC  
    IHSAE001 STEP1      STC18092 IHSAESTC
```

The started tasks are the result of the way the Apache server creates multiple children process to handle the requests. For more information, see Chapter 6, “Scalability and workload management” on page 103.

3.5.7 Verifying that IHS is working

We used the URL <http://wtsc55.itso.ibm.com:8230> to verify that we can access the IHS default home page, which produced the display that is shown in Figure 3-13 on page 58.

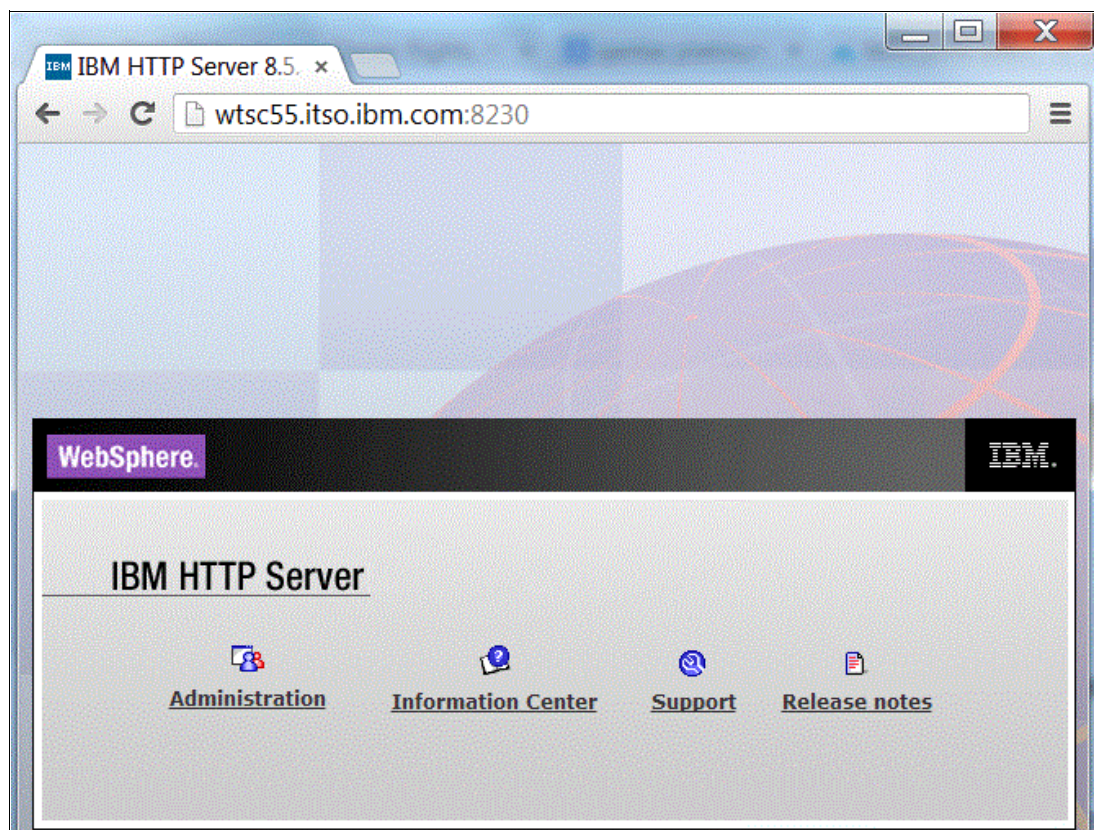


Figure 3-13 Home page of IBM HTTP Server powered by Apache

3.6 Using intermediate symbolic links

When the process that is described in 3.5, “Sample real-world setup process” on page 53 is followed, many separate symbolic links are created from the HTTP server configuration of the HTTP Server to the IBM HTTP Server powered by Apache product code. An example is shown in Example 3-27.

Example 3-27 Symlinks to the product code

```
EDMCAR @ SC55:/ihsconfig/ihs/ihsae001/bin>ls -lrt
total 240
lrwxrwxrwx  1 IHS AE001 IHSRB13      35 Jun 13 21:20 sslstash ->
/ihs/usr/lpp/IHSA/V8R5/bin/sslstash
lrwxrwxrwx  1 IHS AE001 IHSRB13      31 Jun 13 21:20 sidd ->
/ihs/usr/lpp/IHSA/V8R5/bin/sidd
lrwxrwxrwx  1 IHS AE001 IHSRB13      44 Jun 13 21:20 set_attributes.sh ->
/ihs/usr/lpp/IHSA/V8R5/bin/set_attributes.sh
lrwxrwxrwx  1 IHS AE001 IHSRB13      37 Jun 13 21:20 rotatelog ->
/ihs/usr/lpp/IHSA/V8R5/bin/rotatelog
```

Complete the following steps if you need to change the maintenance level of the product code that is used:

1. Stop all IBM HTTP Servers powered by Apache that include symlinks to the directory where the product code is located.

2. Unmount the zFS at the product code directory.
3. Mount the zFS containing the new maintenance level at the product code directory.
4. Start IBM HTTP Servers powered by Apache.

If you have servers running on two z/OS LPARs to provide high availability, all servers must still be stopped on both LPARs if the one product location is being used by both LPARs.

There is a better approach, which is described next.

3.6.1 Setting up an intermediate link

A better approach than the approach that is described in 3.6, “Using intermediate symbolic links” on page 58 is to use a single intermediate symbolic link to point to the product code. This approach is commonly used when WebSphere Application Server cells are set up on z/OS. For more information, see this website:

<https://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100396>

A similar approach can be used for IBM HTTP Server powered by Apache.

On our z/OS LPAR, we created an intermediary symbolic link that is named `/ihsconfig/ihsInstall` by issuing the following commands:

```
cd /ihsconfig
ln -s /usr/lpp/zWebSphereEM1_IHSA/V8R5 ihsInstall
```

The result of these commands is shown in Example 3-28.

Example 3-28 Creating intermediate symbolic link

```
EDMCAR @ SC55:/ihsconfig>ls -lrt
total 96
drwxrwxrwx  6 EDMCAR  SYS1      8192 Jun 14 11:23 ihs
drwxrwxr-x  6 EDMCAR  SYS1      8192 Jun 14 13:04 home
lrwxrwxrwx  1 EDMCAR  SYS1      32 Jun 20 07:30 ihsInstall ->
/usr/lpp/zWebSphereEM1_IHSA/V8R5
```

We then created a second server by using the intermediate symbolic link, as shown in Example 3-29.

Example 3-29 Creating the configuration for IBM HTTP Server powered by Apache

```
su - ihsae001
/ihsconfig/ihsInstall/bin/install_ihs -admin /ihsconfig/ihs/ihsae002 8235
```

The output from this command is shown in Example 3-30.

Example 3-30 Output from creating second server

```
IHSAE001 @ SC55:/ihsconfig/ihs/ihsae002>/ihsconfig/ihsInstall/bin/install_ihs
-admin /ihsconfig/ihs/ihsae002 8235
Copying install directory and creating symlinks...
Updating install paths...
cmd: /ihsconfig/ihsInstall/bin/postinst -i /ihsconfig/ihs/ihsae002 -t install -v
PORT=8235 -v SERVERNAME=wtsc55.itso.ibm.com
Updating permissions for WebSphere Application Server admin console...
```

Example 3-31 shows how the contents of the bin subdirectory of the new server are symbolic links to the intermediate symlink.

Example 3-31 Links to the intermediate symlink

```
EDMCAR @ SC55:/ihsconfig/ihs/ihsae002/bin>ls -lrt
total 176
lrwxrwxrwx  1 IHSAE001 IHSRB13      28 Jun 20 07:34 ab ->
/ihsconfig/ihsInstall/bin/ab
lrwxrwxrwx  1 IHSAE001 IHSRB13      36 Jun 20 07:34 rotatelog ->
/ihsconfig/ihsInstall/bin/rotatelog
lrwxrwxrwx  1 IHSAE001 IHSRB13      36 Jun 20 07:34 preinst.sh ->
/ihsconfig/ihsInstall/bin/preinst.sh
lrwxrwxrwx  1 IHSAE001 IHSRB13      40 Jun 20 07:34 postinstall.sh ->
/ihsconfig/ihsInstall/bin/postinstall.sh
```

With this setup, you can now mount multiple maintenance levels of IBM HTTP Server powered by Apache product code at different mount points. For example, you might have the following maintenance levels mounted:

- ▶ Level N mounted at /usr/lpp/zWebSphereEM1_IHSA /V8R5
- ▶ Level N+1 mounted at /usr/lpp/zWebSphereEM2_IHSA /V8R5

Complete the following steps to change the server to use a new maintenance level:

1. Stop IBM HTTP Servers powered by Apache.
2. Delete the intermediate symlink at /ihsconfig/ihsInstall.
3. Redefine the intermediate symlink to point to the N+1 maintenance level by using `ln -s /usr/lpp/zWebSphereEM2_IHSA/V8R5 ihsInstall`.
4. Start IBM HTTP Servers powered by Apache.

3.7 Maintenance upgrade

IBM HTTP Server powered by Apache V9 is included with z/OS 2.2 and the default installation process results in the product code being part of a zFS that contains other z/OS UNIX components.

If now or in the future you have several Apache V9 servers running on a single LPAR, you might want to consider how to manage changing these servers to a new maintenance level of IBM HTTP Server powered by Apache.

The default approach is to apply maintenance via the standard SMP/E process and then IPL in that updated level of z/OS.

This approach means that all of the IBM HTTP Server powered by Apache servers on that LPAR are upgraded at the same time, which might be acceptable for your organization.

However, if you want to take a more controlled approach in terms of bringing in a new maintenance level of IBM HTTP Server powered by Apache, you might want to configure your environment in a way that allows for such an approach.

Note: For our purposes, we are referring only to Apache servers that use the IBM HTTP Server powered by Apache product code that is supplied with z/OS V2.2. There are other products that run on z/OS that supply their own copy of the IBM HTTP Server powered by Apache product code.

For example, the IBM WebSphere Application Server for z/OS product code includes a corresponding version of IBM HTTP Server powered by Apache.

Apache servers on z/OS that are front ending WebSphere Application Server servers use the product code that is included with the WebSphere Application Server product, which often is stored in a separate zFS.

3.7.1 Gradual maintenance rollout approach

To change one or some IBM HTTP Server powered by Apache servers on a z/OS LPAR to a new maintenance level, you need the following components:

- ▶ A copy of the IBM HTTP Server powered by Apache product code at maintenance level N in its own zFS that is mounted at some directory in the z/OS UNIX environment.
- ▶ A copy of the IBM HTTP Server powered by Apache product code at maintenance level N+1 in its own zFS that is mounted at some directory in the z/OS UNIX environment.

To achieve these maintenance levels, you must copy all of the IBM HTTP Server powered by Apache product code that is under the default `/usr/lpp/ihsa_zos` location; for example, by using the `tar` or `pax` command.

Note: If you copy SMP/E maintained parts from your driving system to your target system, ensure that you verified that all of the affected libraries (which include file systems and PDS and PDSEs) are copied by referring to the appropriate SMP/E reports.

Also ensure that any dependent PTFs (which might be the result of a ++IF RREQ or HOLDS) are copied; otherwise, you risk deploying a partial fix onto your target system.

In addition, ensure that you can determine what PTFs you are deploying in this gradual maintenance rollout. You can get this information by keeping a copy of the SMP/E CSI that is appropriate for what you are rolling out.

Then, mount a new zFS at a new directory location and use the `tar` or `pax` command to restore the contents of the `.tar` or `.pax` file to the new location.

You then have a copy of the IBM HTTP Server powered by Apache product code at maintenance level N in its own zFS mounted at some directory in the z/OS UNIX environment; for example, at the `/ihsv9r0m0` directory.

When you apply maintenance to IBM HTTP Server powered by Apache product code to bring it up to maintenance level N+1, you must perform a similar operation to get a copy of the IBM HTTP Server powered by Apache product code at maintenance level N+1 in its own zFS mounted at some directory in the z/OS UNIX environment; for example, at directory `/ihsv9r0m1`.

The set-up after this process is completed is shown in Figure 3-14.

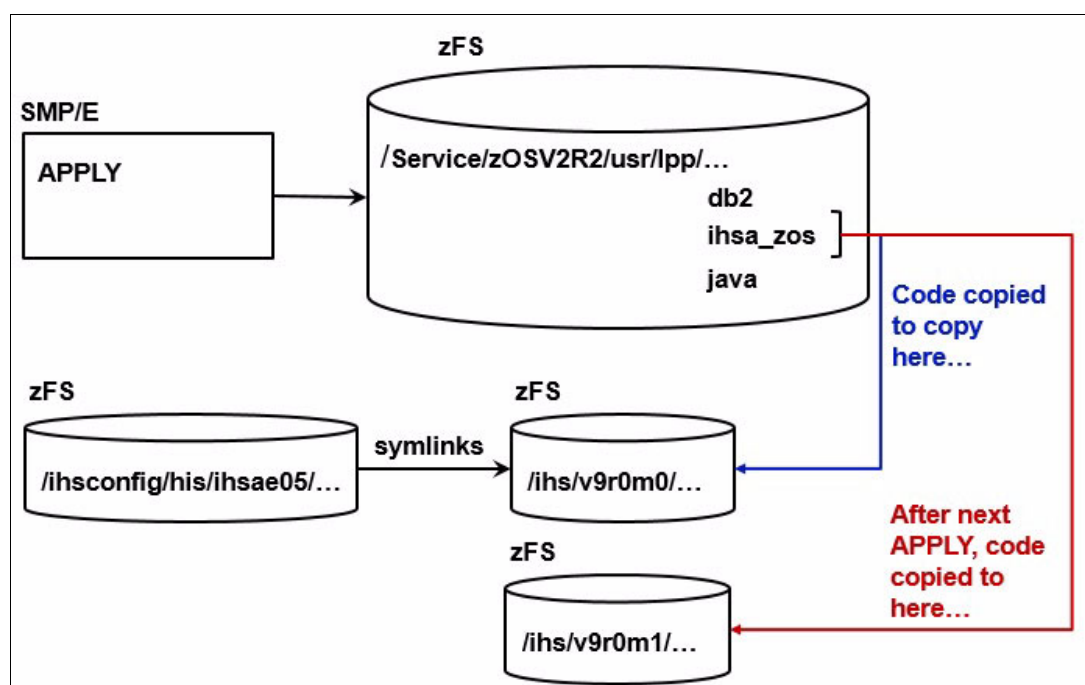


Figure 3-14 Managing different maintenance levels of product code

Figure 3-14 also shows that a defined Apache server includes symlinks pointing to one copy of the IBM HTTP Server powered by Apache product code. In this case, it is the copy at maintenance level v9r0m0.

The result is that you can change individual server instances to use a different maintenance level of the product code by complete the following steps:

1. Stop the server.
2. Change the symbolic link from /ihs/v9r0m to /ihs/v9r0m1.
3. Start the server.

After completing these steps, the symlinks for the defined server point to the copy of the product code in the zFS containing maintenance level v9r0m1.

You can use one of the following approaches to change the symlinks in an individual IBM HTTP Server powered by Apache to a new maintenance level:

- ▶ The symbolic link approach that is described in 3.6.1, “Setting up an intermediate link” on page 59.
- ▶ The new script that is provided with IBM HTTP Server powered by V9 that changes all the symbolic links.

3.7.2 New_install_root shell

Supplied with IBM HTTP Server powered by Apache is a new shell called new_install_root. For more information about this new shell, see this website:

<http://www.ibm.com/support/docview.wss?uid=swg21988561>

The purpose of the new_install_root shell is to change all symbolic links in a defined IBM HTTP Server powered by Apache server instance to a new directory path.

The `new_install_root` shell should be run from the `bin` subdirectory of the IBM HTTP Server powered by Apache product code.

To demonstrate use of the `new_install_root` shell, we set up a second server that is running off a second copy of the IBM HTTP Server powered by Apache product code that includes the following components:

- ▶ A copy of the IBM HTTP Server powered by Apache product code at `/WebSphereEd/ihsv9`.
- ▶ A second copy of the IBM HTTP Server powered by Apache product code at `/WebSphereEd/ihsv9r0m1`.
- ▶ An IBM HTTP Server powered by Apache server instance at `/ihsconfig/ihs/ihsae05` that is built pointing to `/WebSphereEd/ihsv9`.

We started the IHS AE05 server and verified it can access the default home page. We then stopped the IHS AE05 server.

We issued the following commands to change the IHS AE05 instance to point to the alternative copy of the IBM HTTP Server powered by Apache as `/WebSphereEd/v9r0m1` by using the following command:

```
cd /WebSphereEd/ihsv9r0m1/bin
./new_install_root -s /ihsconfig/ihs/ihsae05 -t /WebSphereEd/ihsv9r0m1
```

As the shell runs, it prints every symbolic link that it changes. Part of the output is shown in Example 3-32.

Example 3-32 Partial output from running `new_install_root`

```
'/ihsconfig/ihs/ihsae05/.31bit' -> '/WebSphereEd/ihsv9r0m1/.31bit'
'/ihsconfig/ihs/ihsae05/bin/ab' -> '/WebSphereEd/ihsv9r0m1/bin/ab'
'/ihsconfig/ihs/ihsae05/bin/cgiparse' -> '/WebSphereEd/ihsv9r0m1/bin/cgiparse'

...
'/ihsconfig/ihs/ihsae05/notices' -> '/WebSphereEd/ihsv9r0m1/notices'
'/ihsconfig/ihs/ihsae05/properties' -> '/WebSphereEd/ihsv9r0m1/properties'
'/ihsconfig/ihs/ihsae05/readme' -> '/WebSphereEd/ihsv9r0m1/readme'
postinst: Could not reliably determine the server's fully qualified domain name,
using 127.0.0.1 for ServerName
SERVERROOT=/ihsconfig/ihs/ihsae05
PORT=80
SetupadmUser=EDMCAR
SetupadmGroup=SYS1
SERVERNAME=wtsc55.itso.ibm.com
SHLIBPATH_ENVAR=LIBPATH
PERL=/u/edmcarg/perl
postinst complete
Moving from 9.0.0.0-PI56034 at /WebSphereEd/ihsv9 to 9.0.0.0-PI56034 at
/WebSphereEd/ihsv9r0m1 complete
```

We viewed the files in the IHS AE05 instance at `/ihsconfig/ihs/ihsae05` and verified that the symlinks were updated. The server was restarted and we verified that we can access the default home page.



Administration

IBM HTTP Server powered by Apache can be administered by using the scripts and configuration files that are provided by the product. This chapter describes the procedure to start and stop the server and the way to configure the server and includes the following topics:

- ▶ 4.1, “Running IBM HTTP Server powered by Apache” on page 66
- ▶ 4.2, “Using started tasks” on page 66
- ▶ 4.3, “Using apachectl from the command line” on page 71
- ▶ 4.4, “Integration with WebSphere Application Server” on page 72
- ▶ 4.5, “Configuration” on page 73
- ▶ 4.6, “Monitoring” on page 75
- ▶ 4.7, “Diagnostic tools and information” on page 78
- ▶ 4.8, “Troubleshooting” on page 79
- ▶ 4.9, “Migrating previous versions” on page 79
- ▶ 4.10, “Tracing” on page 80
- ▶ 4.11, “Handling logging” on page 82
- ▶ 4.12, “Macro support” on page 82
- ▶ 4.13, “Conditional controls” on page 83

4.1 Running IBM HTTP Server powered by Apache

The program `apachectl` that is included with IBM HTTP Server powered by Apache for z/OS is used to start and stop a server. You can choose to issue this command interactively from a telnet or OMVS session on the z/OS LPAR or to set up a started task that issues the command.

You often use a started task to manage starting and stopping a server when you want the server to be managed in the same way as any other major component that is running on the z/OS LPAR. A started task approach also fits better if the server is started and stopped by an automation product at IPL times.

Starting the server in a telnet or OMVS session might be useful when you are performing some development work that is associated with your own server.

4.2 Using started tasks

The sample JCL that we used to set up a started task to run an IBM HTTP Server powered by Apache was described in 3.5.6, “Creating a Started Task to run the IHS” on page 57. In this section, we show the commands that can be used to control running the server when it is running as a started task.

4.2.1 Starting the server

The start command that is used in SDSF to start the IHSAE002 server is shown in Example 4-1.

Example 4-1 Issuing start command

```
SDSF DA SC55      SC55      PAG 0  CPU/L/Z
COMMAND INPUT ==> /s ihsae002
NP   JOBNAME  StepName ProcStep JobID
```

The tasks that were started as a result of the use of the start command are shown in Example 4-2.

Example 4-2 Result of start completion

```
SDSF DA SC55      SC55      PAG 0  CPU/L/Z  5/ 4
COMMAND INPUT ==>
NP   JOBNAME  StepName ProcStep JobID      Owner
      IHSAE002 STEP1           STC18243 IHSAE001
      IHSAE002 STEP1           STC18068 IHSAE001
      IHSAE002 STEP1           STC18249 IHSAE001
      IHSAE002 STEP1           STC18250 IHSAE001
      IHSAE002 STEP1           STC18219 IHSAE001
      IHSAE002 IHSAE002 *OMVSEX STC18247 IHSAE001
```

All of the spawned started tasks feature the same name because the original started task included eight characters. If the started task name included seven characters or less, the spawned started tasks each feature a digit appended to their name.

4.2.2 Stopping the server

To stop the server, you must enter a “/” at the Command Input area and then, press Enter. In the System Command Extension area, you next enter the command as shown in Example 4-3.

Example 4-3 Stopping a started task

```
SDSF DA SC55      SC55      PAG 0 CPU/L/Z   4/  2/  0 LINE 1-6 (  
COMMAND INPUT ==> /                                SCR  
NP Exxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
e                                     System Command Extension  
e  
e Type or complete typing a system command, then press Enter.  
e  
e ==> s IHSAE002,action='stop'  
e ==>  
e  
e Place the cursor on a command and press Enter to retrieve it
```

Note: If you enter the command at the Command Input prompt, running the command is unsuccessful because SDSF converts the command to an uppercase format. Also, IBM HTTP Server powered by Apache does not recognize the **action='stop'** part of the command.

After you enter the command, the server stops.

You also can perform a graceful stop of the server. A graceful stop tells the parent process to advise the children to exit after completing all current requests or to exit immediately if no requests are being processed. The parent process exits after all children finalized and exited or the timeout that is specified by the `GracefulShutdownTimeout` is reached. If the timeout is reached, any remaining children are forced to exit, which is done by using the following command:

```
S IHSAE002,action='graceful-stop'
```

4.2.3 Recycling the server to pick up changes

If you change the `httpd.conf` file, it might be possible to have the running server pick up these changes by performing a restart. This restart approach causes each child process to reload the `httpd.conf` file and pick up any changes.

The restart can be done by using one of the following methods:

- ▶ Graceful
- ▶ Restart now

A graceful restart results in the parent process advising the children processes to exit after their current request (or to exit immediately if they are not serving anything). The parent process then rereads its configuration files and reopens its log files. The parent process then replaces each child process as it dies with a child process from the new generation of the configuration, which begins serving new requests immediately.

A restart now results in the parent process to end its children processes without waiting for them to complete any current processes. The parent process then rereads its configuration files and reopens any log files. Then, it creates a set of children processes and continues serving hits. For more information about these approaches, see this website:

<http://httpd.apache.org/docs/2.2/stopping.html>

The started tasks that represent the child processes are not restarted; therefore, the pid of each one does not change.

Example 4-4 shows the use of the **restart** command. If you want to perform a graceful restart, the **graceful** command is used.

Example 4-4 Restarting the server

```
SDSF DA SC55      SC55      PAG 0 CPU/L/Z  4/ 2/ 0 LINE 1-6 (
COMMAND INPUT ==> /                               SCR
NP EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
  e                               System Command Extension
  e
  e Type or complete typing a system command, then press Enter.
  e
  e ==> s ILSAE002,action='restart'
  e ==>
  e
  e Place the cursor on a command and press Enter to retrieve it
```

You see messages in the error_log file that the server restarted.

4.2.4 Modifying command support in V8.5.5

V8.5.5 of IBM HTTP Server powered by Apache provides a new module with which you use standard syntax for issuing commands to a server. To activate this feature, you must add the directive that is shown in Example 4-5.

Example 4-5 Adding directive to support modify commands

```
LoadModule zos_cmds_module modules/mod_zos_cmds.so
```

To stop a server, you can issue the **p ihsae002** command, as shown in Example 4-6.

Example 4-6 Stopping a server

```
SDSF DA SC55      SC55      PAG 0 CPU/L/Z  2/ 2
COMMAND INPUT ==> /p ihsae002
NP  JOBNAME  StepName ProcStep JobID      Owner
    ILSAE002 STEP1      STC18259 ILSAE001
    ILSAE002 STEP1      STC18068 ILSAE001
    ILSAE002 STEP1      STC18249 ILSAE001
    ILSAE002 STEP1      STC18260 ILSAE001
    ILSAE002 STEP1      STC18250 ILSAE001
    ILSAE002 ILSAE002 *OMVSEX STC18262 ILSAE001
```

To perform a graceful stop of the server, enter the following command:

```
F ILSAE002,apl='graceful-stop'
```

To perform a graceful restart of the server, enter the following command:

```
F ILSAE002,appl='graceful'
```

To perform a restart of the server, enter the following command:

```
F ILSAE002,appl='restart'
```

Extraneous options

If you enter an invalid **modify** command, you see output in SDSF that is similar to the output that is shown in Example 4-7.

Example 4-7 Response from issuing invalid command

```
BPXM022E MODIFY SYNTAX ERROR; INVALIDCMD WAS FOUND
WHERE ONE OF THE FOLLOWING WAS EXPECTED:
APPL= TERM= FORCE=
SHUTDOWN= RESTART= DUMP= FILESYS= RECOVER= SUPERKILL=
```

The **modify** command process uses a generic interface that expects the values that are shown in Example 4-12 on page 71. Only the APPL keyword can be used with IBM HTTP Server powered by Apache.

Length of STC name consideration

In “Extraneous options”, it happened that we were using an STC that was eight characters long. This length meant that all of the OMVS STCs that were created featured the same name as the original starting STC.

If your starting STC name is seven characters or less, the OMVS STCs that are created include an extra appended character. For example, we set up a server that is running with the STC name of ILSAE65. In SDSF, you can see the results as shown in Example 4-8.

Example 4-8 SDSF display showing created OMVS STC names

JOBNAME	StepName	ProcStep	JobID	Owner
ILSAE65	ILSAE65	*OMVSEX	STC09071	EDMCAR
ILSAE651	STEP1		STC07168	EDMCAR
ILSAE652	STEP1		STC09063	EDMCAR
ILSAE654	STEP1		STC07172	EDMCAR
ILSAE657	STEP1		STC07169	EDMCAR
ILSAE658	STEP1		STC07196	EDMCAR
ILSAE659	STEP1		STC07191	EDMCAR

When the originating STC name was eight characters long, it might appear that the modify command was being issued to the originating STC. However, it was processed by one of the created OMVS STCs.

Therefore, if you have an originating STC name with seven characters or less, you must identify which of the created OMVS STCs must be targeted with modify commands.

To determine which OMVS STC to use, look in the z/OS syslog for a message when the server is started that is similar to the following example:

```
BPXM023I (EDMCAR) ILS is active. Use jobname ILSAE658 for MVS commands.
```

This message informs you that any modify commands must use F ILSAE658. This message appears in the z/OS syslog only.

If you dynamically restart the server, a new BPXM023I message is issued that advises of the new target OMVS STC.

4.2.5 Displaying version in job log

Complete following steps to display the IBM HTTP Server powered by Apache version in the job log of the started task:

1. Create a Rexx named IHSLEVEL that contains the code that is shown in Example 4-9.

Example 4-9 Rexx code to display Apache version

```
/* REXX */
parse arg ihsDir
say 'ihs dir: ' ihsDir
address tso
call bpxwunix ihsDir"/bin/apachectl -version",,out.
do outx = 1 to OUT.0
  say 'OUT.'outx '=' OUT.outx
  "send '"OUT.outx"' operator(1)"
end outx
exit 0
```

2. Modify the JCL that is used to start the server by adding a first step to start the IHSLEVEL Rexx, as shown in Example 4-10.

Example 4-10 Modified JCL

```
//IHSAE001 PROC ACTION='start',
//          DIR='''/ihsconfig/ihs/ihsae001''',
//          CONF='conf/httpd.conf'
// EXPORT SYMLIST=(IDIR)
// SET IDIR=&DIR
// *-----
//IHSLEVEL EXEC PGM=IKJEFT01
//SYSEXEC DD DISP=SHR,DSN=EDMCAR.Z.REXX
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//SYSTSIN DD *,SYMBOLS=CNVTSYS
//          IHSLEVEL '&IDIR.'
// *-----
//IHS      EXEC PGM=BPXBATCH,
// PARM='SH &IDIR/bin/apachectl -k &ACTION -f &CONF -DNO_DETACH',
// MEMLIMIT=512M
//STDOUT DD PATH='&IDIR/logs/proc.output',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDERR DD PATH='&IDIR/logs/proc.errors',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//          PEND
```

Notice the following line in the JCL in Example 4-10 on page 70:

```
DIR='''/ihsconfig/ihs/ihsae001''',
```

Because the value of the DIR parameter contains lower case characters, triple single quotes must be used so that it is used as a symbol in the JCL.

Sample output

When the server is started, you see messages in the job log that show the Apache version similar to the messages that are shown in Example 4-11.

Example 4-11 Sample Apache version messages

```
15.54.22 STC29431 ---- THURSDAY, 08 DEC 2016 ----
15.54.22 STC29431 IEF695I START IHSAE001 WITH JOBNAME IHSAE001 IS
15.54.22 STC29431 $HASP373 IHSAE001 STARTED
15.54.22 STC29431 +Server version: IBM_HTTP_Server/8.5.5.1 (Unix)
15.54.22 STC29431 +Server built: May 23 2013 00:51:38
15.54.22 STC29431 Jobname Procstep Stepname CPU Time EXCPs
15.54.22 STC29431 IHSAE001 STARTING IHSLEVEL 00:00:00 102
15.54.22 STC29431 IHSAE001 STARTING IHS 00:00:00 102
15.54.22 STC29431 IHSAE001 --None-- *OMVSEX 00:00:00 65
```

Overriding DIR parameter again

If you must override the DIR parameter when the **start** command is issued, triple single quotes must be used if the DIR parameter contains lower-case characters, as shown in the following example:

```
s ihsae001,DIR='''/ihsconfig/ihs/ihsae001'''
```

4.3 Using apachectl from the command line

You can start a server from the command line in a z/OS environment by logging on to the z/OS LPAR through Telnet or by using the OMVS command from ISPF to get to a command prompt. The **apachectl** command to use for starting, stopping, or restarting the server features the following format:

```
./apachectl -k <option>
```

4.3.1 Starting the server

To start the IHSAE002 server, issue the commands that are shown in Example 4-12.

Example 4-12 Starting a server from the command line

```
su - ihsae002
cd /ihsconfig/ihs/ihsae002/bin
./apachectl -k start
```

No message is displayed in response to the start command in the session. Looking in SDSF, you can see that the server started because several started tasks are displayed, as shown in Example 4-13.

Example 4-13 The IHSAE002 server is now running

SDSF	DA	SC55	SC55	PAG	0	CPU/L/Z	3/ 2
COMMAND INPUT ==>							
NP	JOBNAME	StepName	ProcStep	JobID	Owner		
	IHSAE001	STEP1		STC18243	IHSAE001		
	IHSAE001	STEP1		STC18259	IHSAE001		
	IHSAE001	STEP1		STC18068	IHSAE001		
	IHSAE001	STEP1		STC18260	IHSAE001		
	IHSAE001	STEP1		STC18250	IHSAE001		

The name of the started tasks is the user ID that we were logged on as when the **apachectl** command was issued. It is not the logical name of the server.

4.3.2 Stopping the server

To stop the server, issue one of the following commands, depending on what style of shutdown you want to perform:

```
./apachectl -k stop
./apachectl -k graceful-stop
```

4.3.3 Restarting the server

To restart the server, issue one of the following commands, depending on what style of restart you want to perform:

```
./apachectl -k restart
./apachectl -k graceful
```

4.3.4 Mix and match

You can use the **modify** commands (as described in 4.2, “Using started tasks” on page 66) and the **apachectl** command from the command line to perform actions on a server. When the **apachectl** command is used, the server on which it acts is the one corresponding to the configuration directory from which you are issuing the command.

4.4 Integration with WebSphere Application Server

If you are using WebSphere Application Server, you can register your IBM HTTP Servers powered by Apache with the WebSphere cell if they are defined in a managed node. By using this configuration, the WebSphere Application Server administration console can be used to stop and start the server. For more information, see this website:

<https://ibm.biz/Bdr6yb>

4.5 Configuration

IBM HTTP Server powered by Apache features a main default configuration file that is named `httpd.conf`. A sample configuration file that is named `httpd.conf.default` also is available and is a copy of the original `httpd.conf` file. Both files can be found in the `/conf` subdirectory where you set up the server. It is the `httpd.conf` file in which you must make configuration changes to have it perform in the way you require.

IBM HTTP Server powered by Apache uses directives that are native to the original Apache HTTP Server and directives that are available because more modules and features were added by IBM.

Explaining all possible uses of the available directives is beyond the scope of this paper. Many examples of how to configure the Apache HTTP Server to handle user requirements can be found by searching the Internet.

This section reviews some of the key aspects of the `httpd.conf` file that you are more likely to modify.

4.5.1 Listen directive

The `Listen` directive specifies the TCP/IP port on which the IBM HTTP Server powered by Apache listens. If you want to change this port, edit the `Listen` directive in the `httpd.conf` file, as shown in Example 4-14.

Example 4-14 Listen directive in the httpd.conf file

```
# Listen: Allows you to bind the web server to specific IP addresses
# and/or ports, in addition to the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent the web server from accepting connections on all interfaces
# (0.0.0.0)
#
# Change this to "Listen 0.0.0.0:port" to restrict the server to
# IPv4.
#
Listen 8237
```

4.5.2 Virtual hosting

One of the most important features of IBM HTTP Server powered by Apache is the `VirtualHost` directive. By using this feature, any number of directives that are to apply can be associated to requests that match that directive. One server also can handle requests for any number of logical domains. To enable this feature, you must specify at least the server name of your site and the port.

The `VirtualHost` directive can include any other directives that apply only to that specific site. One of these directives is `DocumentRoot`, which defines the folder that contains the documents that your site serves to clients.

In case your IBM HTTP Server powered by Apache hosts more than one site, you might find it necessary to set separate document root folders for each site. For example, you can set an IBM HTTP Server powered by Apache to host two websites that are named `www.ibmitsosite1.com` and `www.ibmitsosite2.com`. The virtual host definitions might be configured as shown in Example 4-15.

Example 4-15 Virtual hosts definition for two sites that are hosted by a single server

```
Listen 80
Listen 443
NameVirtualHost ibmitsosite1:80
NameVirtualHost ibmitsosite2:80
NameVirtualHost ibmitsosite:80
NameVirtualHost ibmitsosite:443
<VirtualHost www.ibmitsosite1.com:80>
    DocumentRoot /www/ibmitsosite1
    DirectoryIndex index.html index.htm
    ErrorDocument 404 /www/ibmitsosite1/error404_1.html
    ErrorDocument 500 /www/ibmitsosite1/error500_1.html
    ErrorLog logs/ibmitsosite1_80_error.log
    TransferLog logs/ibmitsosite1_80_access.log
    LogLevel error
</VirtualHost>
<VirtualHost www.ibmitsosite2.com:80>
    DocumentRoot /www/ibmitsosite2
    DirectoryIndex index.html index.htm
    ErrorDocument 404 /www/ibmitsosite2/error404_2.html
    ErrorDocument 500 /www/ibmitsosite2/error500_2.html
    ErrorLog logs/ibmitsosite2_80_error.log
    TransferLog logs/ibmitsosite2_80_access.log
    LogLevel info
</VirtualHost>
<VirtualHost www.ibmitsosite.com:80>
    DocumentRoot /www/ibmitsosite_80_to_1
    DirectoryIndex index.html index.htm
    Redirect permanent / http://www.ibmitsosite1.com/
</VirtualHost>
<VirtualHost www.ibmitsosite.com:443>
    DocumentRoot /www/ibmitsosite_443_to_2
    DirectoryIndex index.html index.htm
    Redirect permanent / http://www.ibmitsosite2.com/
</VirtualHost>
```

This example assumes that TCP/IP in your IBM HTTP Server powered by Apache host resolves both site names and the `www.ibmitsosite.com` that is redirected to `http://www.ibmitsosite1.com` if the request is on the port 80, and to `http://www.ibmitsosite2.com` if the request is on port 443. The log levels are set to different levels and files for each site.

4.6 Monitoring

IBM HTTP Servers powered by Apache can be monitored by using the methods that are described in this section.

4.6.1 SDSF

If you are running your servers as started tasks, a good first check is to see whether the started tasks are running if you suspect a problem.

4.6.2 Checking pid and log files

Common files to check on your server include the pid and log files. These files are in the log subdirectory of the configuration directory of the server. In our case, the `/ihs/config/ihs/ihsae002/logs` directory was used. Check the following files:

- ▶ `httpd.pid`: This process identity file contains the process number of the initial parent process of the server. This file is refreshed when the server is started and you can use it to search for the server process.
- ▶ `error_log`: This file is where the server logs alert, notice, warning, and error messages are stored. You can use this log to view events, such as the hours that the server was last restarted.
- ▶ `access_log`: This file is where the server logs records of all requests processed are stored. You can use this log to search for information, such as how many 404 HTML response code messages were received during a specific time. Also, this file contains by default the IP addresses of the clients for which requests are processed.

Log rotation

The `error_log` and `access_log` files are not rotated by default. Therefore, their size might grow to adversely affect your environment. One way to rotate the logs is to stop the server, move the log files to another location, and then start the server. The problem with this method is that you must restart the server, which means that your clients cannot access it.

To avoid this situation, configure rotation for this log files by using piped logs. IBM HTTP Server powered by Apache can write error and access log files through a pipe to another process. This feature is inherited from Apache. To configure piped logs, you must use a program called `rotatelogs` that is in the `bin` folder of your IBM HTTP Server powered by Apache installation path. Example 4-16 shows a way to rotate the `access_log` every one hour.

Example 4-16 Rotating the `access_log` every 12 hours

```
CustomLog "|/ihsam001/bin/rotatelogs /ihsam001/logs/access_log 3600" common
```

The format of the access log file can be customized by using different percent directives that you can add to the default `LogFormat` directive. Example 4-17 on page 76 shows the default log format configuration and one output line from the `access_log` that indicates the following information:

- ▶ The IP of the client request (as in the `%h` directive).
- ▶ The first hyphen that indicates the identity of the client machine, which in our case is not available (as in the `%l` directive).
- ▶ The second hyphen that indicates the identity of the request user, which in our case is not available (as in the `%u` directive).

- ▶ The time that the request was received by the server (as in the `%t` directive).
- ▶ The method that is used by the client, requested resource, and protocol type (as in the `\"%r\"` directive).
- ▶ The status code that is sent back by the server to the client (as in the `%>s` directive).
- ▶ The size of the object that is returned to the client (as in the `%b` directive).

Example 4-17 Default access_log format and output

```
IHSAM001 @ SC55:/ihsam001>cat ./conf/httpd.conf | grep LogFormat
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
IHSAM001 @ SC55:/ihsam001>cat ./conf/httpd.conf | grep access_log
CustomLog "|/ihsam001/bin/rotatelogs /ihsam001/logs/access_log 60000" common
IHSAM001 @ SC55:/ihsam001>tail -1 ./logs/access_log.123.123.123 - -
[18/Jun/2013:16:18:54 -0400] "GET /images/administration.gif HTTP/1.1" 200 223
```

4.6.3 Server status

The default `httpd.conf` file contains the lines that are shown in Example 4-18.

Example 4-18 Directives to allow for server status reports

```
# Allow server status reports generated by mod_status,
# with the URL of http://servername/server-status
# Change the ".example.com" to match your domain to enable.
#
<IfModule mod_status.c>
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
```

The previous directives provide a URL that you can use to display the current state of the server. It shows the requests that are being processed, available threads, and so on.

However, the default setup does not allow anyone to use it. The simplest change allows anyone to use the URL. The required change is shown in Example 4-19.

Example 4-19 Allowing everyone access to the server status URL

```
# Allow server status reports generated by mod_status,
# with the URL of http://servername/server-status
# Change the ".example.com" to match your domain to enable
#
<IfModule mod_status.c>
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Allow from all
```

A sample of the output produced is shown in Figure 6-5 on page 115.

4.6.4 Server status by using the modify command

For more information about setting up the IBM HTTP Server powered by Apache so that you can view the status of requests being processed by issuing a request from a browser, see 4.6.3, “Server status” on page 76.

APAR PI24990 provides the ability to use a modify command to display this same type of information so that it is displayed in the z/OS system log. The output is not displayed in the STCs that run IBM HTTP Server powered by Apache.

The APAR also provides an option to reset the statistic counters. For more information about the APAR, see this website:

<http://www.ibm.com/support/docview.wss?uid=isg1PI24990>

The support for this APAR will be available in V8.5.5.4 of the IBM HTTP Server powered by Apache for z/OS.

To display server stats information, issue the following command:

```
F IHS AE658,APPL='stats'
```

Example 4-20 shows the output of this command.

Example 4-20 Sample of output displayed after issuing stats command

```
F IHS AE658,APPL='stats'
BPXM023I (EDMCAR) 280
IHS stats: hostname: wtsc69.itso.ibm.com ppid: 67240308
Interval: 84s Requests: 10 bytes: 241815
%busy: 0 (0,50,600)
rdy 50 bsy 0 rd 0 wr 0 log 0 dns 0 cls 0 ka na
```

To reset the server statistics information, use the **resetstats** or **restats** keyword, as shown in the following command:

```
F IHS AE658,APPL='restats'
```

Example 4-21 shows a sample of the output that is produced by this command.

Example 4-21 Sample of resetstats command output

```
BPXM023I (EDMCAR) 282
IHS stats: hostname: wtsc69.itso.ibm.com ppid: 67240308
Interval: 7m Requests: 10 bytes: 241815
%busy: 0 (0,50,600)
rdy 50 bsy 0 rd 0 wr 0 log 0 dns 0 cls 0 ka na
```

4.6.5 Thread usage

The **mpmstats** module (if enabled) writes information periodically about thread usage by the server. For more information about this module and the information it provides, see this website:

https://publib.boulder.ibm.com/httperv/manual70/mod/mod_mpmstats.html

For more information about how the information that is produced by the **mpmstats** module can be written to SMF records, see 8.3.4, “Enabling for subtype 13” on page 134.

4.7 Diagnostic tools and information

IBM provides a utility package that is named `ihsdiag` for capturing diagnostic information about IBM HTTP Server powered by Apache. This tool helps you investigate problems and send diagnostic information to IBM support.

The `ihsdiag` tool can be downloaded from the following website:

<https://www.ibm.com/support/docview.wss?uid=swg24008409>

The package contains the following items:

- ▶ Documentation for configuring IBM HTTP Server powered by Apache and the operating system for problem determination.
- ▶ Diagnostic modules to load into IBM HTTP Server powered by Apache to capture first failure information.
- ▶ Diagnostic tools for gathering information about problem symptoms, including child process crashes, hang conditions, high CPU conditions, and startup failures.
- ▶ IBM HTTP Server powered by Apache performance tuning information.
- ▶ IBM HTTP Server powered by Apache Q&A document

You must install the tool on your system. A prerequisite is that the `gzip` package is installed. The tool is started as a Java component and must run under an IBM Java Runtime of 1.4.2 or later. The tool creates a directory that contains a time stamp in the name. The gathered information is saved in that directory.

The `ihsdiag` tool accepts the following tasks:

- ▶ `DescribeSingleProcess`
- ▶ `DescribeAllProcesses`
- ▶ `ListAllProcesses`
- ▶ `GatherCrashDoc`
- ▶ `GatherHangDoc`
- ▶ `GatherHighCpuDoc`
- ▶ `CheckPlatform`
- ▶ `DescribeConfig`
- ▶ `ParseNetTrace`

Example 4-22 shows an output of the `ihsconfig` tool.

Example 4-22 DescribeConfig task output of ihsdiag tool

```
HAIMO @ SC55:/ihsam001/ihsdiag-1.4.16>java -jar ./ServerDoc.jar DescribeConfig ihsam001
Web server version: 8.5.5.1
Reports, log files, and configuration files have been saved to directory
  ServerConfig.201306191554
If you have additional log files or configuration files, copy them there
before packing up the directory.
Web server log and conf files other than the default will have to be
copied manually.
WebSphere plug-in conf and log files will have to be copied manually.
```

Hint for packing up the directory:

```
pax -wo to=IS08859-1 -f ServerConfig.201306191554.tar ServerConfig.201306191554
compress ServerConfig.201306191554.tar
```

4.8 Troubleshooting

Complete the following steps to troubleshoot your IBM HTTP Server powered by Apache installation:

1. Check the error log of the server for problems. Look for lines that contain strings, such as {alert}, [error], or [warn] because this type of messages might indicate the cause of the problem.
2. Check the IHS Diagnostic Tools and Information package that is described in 4.7, “Diagnostic tools and information” on page 78 for more diagnostic information, and the MustGather steps for some problems. If you open an IBM Problem Management Record (PMR) for IBM HTTP Server powered by Apache, you are prompted for the output of the `ihdiag` tool.
3. Ensure that the latest IHS fix level is installed. In most cases, the problem you encounter was resolved in a fix that is available for downloading. For more information about IBM HTTP Server powered by Apache recommended updates, see this website:

<http://www.ibm.com/support/docview.wss?rs=177&context=SSEQTJ&uid=swg27005198>

4. Check the following IBM HTTP Server powered by Apache support page for Technotes:

<http://www.ibm.com/software/webservers/httpservers/support/>

By following the first step of this procedure, you often can find the root cause of the problem. Example 4-23 shows the output in the `error_log` for a common problem, namely trying to access a file that does not exist.

Example 4-23 Error output in error_log

```
[Tue Jun 18 16:19:02 2013] [error] [client 9.123.123.123] File does not exist:
/ihsam001/htdocs/z
```

You can find two configuration issues. For more information about solving these issues, see this website:

<https://ibm.biz/Bdr6Sq>

4.9 Migrating previous versions

No mechanism is supplied by IBM to migrate the configuration of an IBM HTTP Server powered by Apache to a new version. Complete the following steps to migrate to a new version:

1. Install the product code of the new version.
2. Use the new version of the product to create a server.
3. Modify the new `httpd.conf` file with the same changes you made to your server.
4. Make any other changes to the new configuration that you made to the server configuration.

You can use a command, such as `diff`, to display the differences between your current `httpd.conf` file and the new `httpd.conf.default`.

Typically there are not many changes in the directives and modules between versions, but there can be some changes. Review any changes in the new version and determine whether these changes affect your server. For more information about how two new modules called `mod_authnz_saf` and `mod_authnz_ldap` replace similar modules that are used in older versions, see this website if you were migrating to Version 8.5:

<https://ibm.biz/Bdr6ke>

4.10 Tracing

IBM HTTP Server powered by Domino provided a method to trace the way it processed a request.

IBM HTTP Server powered by Apache V9 now also provides a tracing capability. This tracing capability is provided by Apache V2.4 upon which this version is built.

The tracing capability provides a way to trace the way the server processes a request, which can be useful when the server does not process the request in the way you expect.

4.10.1 Information about tracing

The new tracing capability is controlled by new values that can be supplied on the `LogLevel` directive. The new values that can be used all start with the value “trace”, as shown in Figure 4-1.

Level	Description	Example
emerg	Emergencies - system is unusable.	"Child cannot open lock file. Exiting"
alert	Action must be taken immediately.	"getpwuid: couldn't determine user name from uid"
crit	Critical Conditions.	"socket: Failed to get a socket, exiting child"
error	Error conditions.	"Premature end of script headers"
warn	Warning conditions.	"child process 1234 did not exit, sending another SIGHUP"
notice	Normal but significant condition.	"httpd: caught SIGBUS, attempting to dump core in ..."
info	Informational.	"Server seems busy. (you may need to increase StartServers, or Min/MaxSpareServers)..."
debug	Debug-level messages	"Opening config file ..."
trace1	Trace messages	"proxy: FTP: control connection complete"
trace2	Trace messages	"proxy: CONNECT: sending the CONNECT request to the remote proxy"
trace3	Trace messages	"openssl: Handshake: start"
trace4	Trace messages	"read from buffered SSL brigade, mode 0, 17 bytes"
trace5	Trace messages	"map lookup FAILED: map=rewritemap key=keyname"
trace6	Trace messages	"cache lookup FAILED, forcing new map lookup"
trace7	Trace messages, dumping large amounts of data	" 0000: 02 23 44 30 13 40 ac 34 df 3d bf 9a 19 49 39 15 "
trace8	Trace messages, dumping large amounts of data	" 0000: 02 23 44 30 13 40 ac 34 df 3d bf 9a 19 49 39 15 "

Figure 4-1 Settings to use for enabling tracing

For more information about the new tracing-related directives, see this website:

<http://httpd.apache.org/docs/current/mod/core.html#loglevel>

For more information about how to perform tracing, see this website:

<http://people.apache.org/~trawick/AC2014-Debug.pdf>

Trace output is written to the error_log file.

4.10.2 Limitation

A limitation of the new tracing capability is that it cannot be started, stopped, or changed dynamically. Any changes that are related to the tracing directives require a stop and start of the server.

4.10.3 Some examples

Assume that the level of tracing is set to warn in the LogLevel directive.

We now want to trace how the server is processing requests that start with /banking/deposits. Example 4-24 shows how we code directives to achieve this goal.

Example 4-24 Tracing requests that match a URL

```
# Maximum trace for this URI
<Location /misbehavingApplication/>
    LogLevel trace8
</Location>
```

Next, we want to trace all requests that are coming from a certain TCP/IP address. How the directives are coded for this trace is shown in Example 4-25.

Example 4-25 Tracing requests from a specific TCP/IP address

```
# Maximum trace from a specific client
<if "%{REMOTE_ADDR} == '10.1.2.3'>
    LogLevel trace8
</if>
```

The tracing that is produced in Example 4-24 and Example 4-25 show output from all modules that were involved with tracing the request.

How to code the directives so that only actions that are performed by the rewrite module are output is shown in Example 4-26. IBM HTTP Server powered by Domino was unable to perform this tracing.

Example 4-26 Tracing only the actions taken by the rewrite module

```
# We're debugging just the mod_rewrite module here
<LocationMatch ^/app2/controller.do$>
    LogLevel rewrite:trace8
</Location>
```

We enabled tracing in our IBM HTTP Server powered by Apache by using the approach that is shown in Example 4-25 and accessed the default home page. A small sample of the trace output produced is shown in Example 4-27.

Example 4-27 Example of trace output

```
[Tue Jan 20 16:52:09.032625 2015] [core:trace4] [pid 131234:tid 2725491413163704327]
util_expr_eval.c(835): [client 9.190.237.30:54034] Evaluation of expression from
/ihsconfig/ihs/ihsae009/conf/httpd.conf:899 gave: 1, referer:
http://wtsc55.itso.ibm.com:8229/
```

```
[Tue Jan 20 16:52:09.032636 2015] [authz_core:debug] [pid 131234:tid 2725491413163704327]
mod_authz_core.c(799): [client 9.190.237.30:54034] AH01626: authorization result of Require
all granted: granted, referer: http://wtsc55.itso.ibm.com:8229/
[Tue Jan 20 16:52:09.032640 2015] [authz_core:debug] [pid 131234:tid 2725491413163704327]
mod_authz_core.c(799): [client 9.190.237.30:54034] AH01626: authorization result of
<RequireAny>: granted, referer: http://wtsc55.itso.ibm.com:8229/
[Tue Jan 20 16:52:09.032674 2015] [core:trace3] [pid 131234:tid 2725491413163704327]
request.c(236): [client 9.190.237.30:54034] request authorized without authentication by
access_checker_ex hook: /images/ihs/support.gif, referer: http://wtsc55.itso.ibm.com:8229/
[Tue Jan 20 16:52:09.032679 2015] [charset_lite:trace2] [pid 131234:tid
2725491413163704327] mod_charset_lite.c(333): [client 9.190.237.30:54034]
should_translate_request: r->handler=(null), referer: http://wtsc55.itso.ibm.com:8229/
```

4.11 Handling logging

Apache always supplied a program that is named `rotatelogs`, which supports rotating logs based on a time interval or maximum size of the log.

In Apache 2.4, the following enhancement were added to `rotatelogs`:

- Custom post-rotate script

`Rotatelogs` provides a new option `-p`, which can be used to specify a name of a program that is called when a log file is created. The name of the previous log file is passed as a parameter to the program specified.

For more information, see this website:

<http://httpd.apache.org/docs/current/programs/rotatelogs.html>

By using this option, a process can be set up to better manage log files as they are produced.

- Circular list of output files

The normal processing of log files includes IBM HTTP Server powered by Apache running `rotatelogs` according to how you configure it, which creates a log file. However, you then must set up some mechanism to manage these old log files.

An alternative approach is to use the new `-n` option to specify how many log files are kept by the server. If you specify `"-3"` for example, the server creates three log files and cycles through writing output to each one in turn only.

4.12 Macro support

IBM HTTP Server powered by Apache V9 inherits the support that Apache 2.4 has for the Macro directive. This capability is provided by a third-party module.

An example of where this macro capability is useful is if you are running an IBM HTTP Server powered by Apache to support several virtual hosts. In this scenario, you might have large a `httpd.conf` file. The file is large because each virtual host features many directives with different values.

By using this marco capability, you can simplify the way your httpd.conf file is set up.

An example of the use of the macro directive is shown in Figure 4-2. In the example, a macro that is named VHost was defined and expects three parameters. The macro then contains several directives that referenced the passed parameters. The next three lines after the macro definition show that the macro was called three times, which resulted in three defined VirtualHost stanzas.

```
## Define a VHost Macro for repetitive configurations

<Macro VHost $host $port $dir>
  Listen $port
  <VirtualHost *: $port>

    ServerName $host
    DocumentRoot $dir

    <Directory $dir>
      # do something here...
    </Directory>

    # limit access to intranet subdir.
    <Directory $dir/intranet>
      order deny,allow
      deny from all
      allow from 10.0.0.0/8
    </Directory>
  </VirtualHost>
</Macro>

## Use of VHost with different arguments.

Use VHost www.apache.org 80 /projects/apache/web
Use VHost www.perl.com 8080 /projects/perl/web
Use VHost www.ensmp.fr 1234 /projects/mines/web
```

Figure 4-2 Example showing use of maro directive

4.13 Conditional controls

IBM HTTP Server powered by Apache V9.0 is built on Apache 2.4 and supports the following directives that were added by Apache 2.4:

- ▶ if
- ▶ elseif
- ▶ else

For more information about these directives, see this website:

<http://publib.boulder.ibm.com/httperv/manual24/mod/core.html#if>

IBM HTTP Server powered by Apache administrators always had a requirement to manage conditional type requirements. Before these new directives, the common approach was to use ReWrite directives. However, the use of a ReWrite directive can result in complex syntax that is difficult to interpret.

The <if> and other new directives can be used in all contexts and provide a clearer way to implement conditional requirements.

Examples of how to use this new capability are shown in Figure 4-3.

```
# Compare the host name to example.com and redirect to www.example.com if it
matches
<If "%{HTTP_HOST} == 'example.com'">
    Redirect permanent "/" "http://www.example.com/"
</If>

# Force text/plain if requesting a file with the query string contains
'forcetext'
<If "%{QUERY_STRING} =~ /forcetext/">
    ForceType text/plain
</If>

# Only allow access to this content during business hours
<Directory "/foo/bar/business">
    Require expr %{TIME_HOUR} -gt 9 && %{TIME_HOUR} -lt 17
</Directory>
```

Figure 4-3 Examples of conditional control



Migration

This chapter provides guidance about migrating from IBM HTTP Server powered by Domino to IBM HTTP Server powered by Apache.

This chapter includes the following topics:

- ▶ 5.1, “Planning your migration” on page 86
- ▶ 5.2, “Migration guidance” on page 87
- ▶ 5.3, “Migrating Library Server” on page 97

5.1 Planning your migration

As a wise old computer programmer once advised, there are only three key factors when it comes to performing a successful migration:

- ▶ Planning
- ▶ Planning
- ▶ Planning

Therefore, a migration plan must be developed first to migrate from IBM HTTP Server powered by Domino to IBM HTTP Server powered by Apache.

5.1.1 Migration plan

A plan outlines the steps that must be performed to migrate to IBM HTTP Server powered by Apache. The plan helps you to identify the following issues:

- ▶ What needs to be done.
- ▶ Who is involved.
- ▶ Who is responsible for various tasks.
- ▶ How long it takes.

If you have only a few HTTP Servers powered by Domino that are infrequently used and that usage is basic, your plan likely is simple to prepare and run. However, if you have many HTTP Servers powered by Domino that use many GWAPI programs and various security functions, your plan takes longer to prepare and is more detailed.

Determining what must be done

Your objectives include the following areas:

- ▶ Identify which IBM HTTP Servers powered by Domino are in use and on what z/OS LPARs they run.
- ▶ Determine from where the IBM HTTP Server powered by Apache product code will be obtained.
- ▶ Find information sources, such as this document, that can assist your understanding of the new product.
- ▶ Determine how your IBM HTTP Servers powered by Domino are being used, including the following examples:
 - Is security being used?
 - Does the server handle multiple hosts?
 - Does the server listen on multiple ports?
 - Are GWAPI modules being used, and do you have the source code for these modules?
 - Is it set up to produce SMF records?
 - Is it running in scalable mode?
- ▶ Plan the installation and configuration of new servers.
- ▶ Plan how you to switch over from old to new servers:
 - Will this be a “big bang” approach, doing all servers at the same time?
 - If running in a sysplex, you might bring in new servers on one LPAR first.

Who is to be involved

This aspect includes the following considerations:

- ▶ Identifying the users of these servers so you can inform them that a change to IBM HTTP Server powered by Apache is being planned.
- ▶ Identifying owners of applications that are accessed by the current servers.
- ▶ Identifying technical staff who must be involved in the migration process.

Responsibility for various tasks

After you identify the various groups who are involved your plan, you must clearly identify their roles and responsibilities.

How long it takes

Your plan should include a timeline that shows when changes will be made and how long tasks in the plan take.

5.2 Migration guidance

This section describes the following common aspects of a migration to IBM HTTP Server powered by Apache:

- ▶ Scalable mode
- ▶ SMF records
- ▶ Server home directory
- ▶ Ports
- ▶ Virtual hosts
- ▶ Security
- ▶ Logging
- ▶ URL and file mapping directives
- ▶ WebSphere Application Server plug-in
- ▶ Timeouts
- ▶ Caching
- ▶ ASCII/EBCDIC considerations
- ▶ GWAPI
- ▶ Reverse Proxy
- ▶ Clean up PARMLIB

5.2.1 Scalable mode

If you are running servers in scalable mode, see Chapter 6, “Scalability and workload management” on page 103, which compares scalable mode to the WLM classification support that is provided in V8.5.5 of IBM HTTP Server powered by Apache.

5.2.2 SMF records

If your servers are set up to produce SMF records, see Chapter 8, “System Management Facilities support in IHS V8.5.5” on page 131, which compares the SMF records written by the two products.

5.2.3 Server home directory

The server home directory directives in IBM HTTP Server powered by Domino and IBM HTTP Server powered by Apache are slightly different. IBM HTTP Server powered by Domino `InstallPath` and `ServerRoot` directives are replaced in IBM HTTP Server powered by Apache with the `ServerRoot` directive. There is no such directive as `InstallPath` in IBM HTTP Server powered by Apache. Example 5-1 shows how these directives are used in IBM HTTP Server powered by Domino.

Example 5-1 InstallPath and ServerRoot

```
# IBM HTTP Server powered by Domino InstallPath directive:
# Set this to point to the server install path
# Default: /Z1DRC1/usr/lpp/internet
# Syntax: InstallPath <path>
InstallPath /Z1DRC1/usr/lpp/internet

# IBM HTTP Server powered by Domino ServerRoot directive:
# Set this to point to the directory where you unpacked this
# distribution, or wherever you want HTTPd to have its "home".
# By default this directory is located in the install path
# specified by the InstallPath directive.
# Default: server_root
# Syntax: ServerRoot <path>
ServerRoot server_root
```

Example 5-2 shows the corresponding directive and how it is used in IBM HTTP Server powered by Apache.

Example 5-2 ServerRoot directive

```
# IBM HTTP Server powered by Apache ServerRoot directive:
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
# Do NOT add a slash at the end of the directory path.
ServerRoot "/ihsconfig/ihs/ihsam001"
```

The IBM HTTP Server powered by Domino `Welcome` equivalent in IBM HTTP Server powered by Apache is the `DirectoryIndex` directive. IBM HTTP Server powered by Domino `Welcome index.html` becomes IBM HTTP Server powered by Apache `DirectoryIndex index.html`.

5.2.4 Ports

IBM HTTP Server powered by Apache supports the use of multiple SSL and non-SSL ports. You can use only non-SSL ports or only SSL ports. For each port, you want IBM HTTP Server powered by Apache to listen on, you define a `Listen` directive. For example, use the `Listen 9080` directive to have the server listen on port 9080.

SSL

To set up a port to use SSL, you must configure a `VirtualHost` stanza that contains an `SSLEnable` directive. It is a good idea to use certificates stored in RACF, although you can use keystore files.

IBM HTTP Server powered by Domino requires configuring every SSLCipherSpec that you need, whereas IBM HTTP Server powered by Apache enables all SSLCipherSpec by default. If you want to use only certain ciphers, you can use a combination of SSLCipherSpec and SSLProtocolDisable directives. Example 5-3 shows a sample configuration that provides the following actions:

- ▶ Server listens for HTTP on ports 81 and 87.
- ▶ Server listens on port 444. This SSL port is associated with the virtual host `www.site1.com`. The certificate is stored in RACF with a label of `site1_SAF_ring`, and SSL V2 is disabled.
- ▶ Server listens on port 447. This SSL port is associated with the virtual host `www.site2.com`. The certificate is stored in a keystore file. Only some SSL ciphers are enabled and some are disabled.

Example 5-3 IBM HTTP Server powered by Apache port definition and virtual host SSL configuration

```
Listen 81
Listen 87
Listen 444
Listen 447
<VirtualHost hostname1:444>
    ServerName www.site1.com:444
    SSLEnable
    Keyfile /saf site1_SAF_ring
    SSLProtocolDisable SSLv2
</VirtualHost>
<VirtualHost hostname1:447>
    ServerName www.site2.com:447
    SSLEnable
    SSLProtocolDisable SSLv2 TLSv1
    SSLCipherSpec 3A
    SSLCipherSpec 35
    SSLCipherSpec 34
    Keyfile /ihsconfig/ihs/ihsam001/ssl/site2.kdb
    SSLServerCert KDBLabel
</VirtualHost>
```

5.2.5 Virtual hosts

Both versions of IBM HTTP Server support multiple virtual hosts.

The DGW approach is to add the domain name or IP address to the end of directives, such as `Exec`, `Fail`, `Map`, `Pass`, and `Redirect`. For more information about the DGW approach, see this website:

<https://ibm.biz/BdrUMj>

IBM HTTP Server powered by Apache uses the underlying Apache approach. For more information, see this website:

<http://httpd.apache.org/docs/current/vhosts/>

The Apache approach is to use the `VirtualHost` directive to identify a virtual host. Then, you define any other directives within that stanza that are to apply to that virtual host.

To migrate from DGW to IBM HTTP Server powered by Apache, you must identify directives in DGW that include domain names or IP addresses that are associated with them. Then, you set up `VirtualHost` type directives to use for the identified domain name or TCP/IP addresses.

For more information about the use of multiple VirtualHost directives, see 4.5.2, “Virtual hosting” on page 73.

5.2.6 Security

As you migrate your security aspects from IBM HTTP Server powered by Domino to IBM HTTP Server powered by Apache, consider the following issues:

- ▶ User ID running your server
- ▶ LDAP authentication
- ▶ Authentication and authorization
- ▶ Key files

The default user ID that is running your server configuration in IBM HTTP Server powered by Domino is user ID PUBLIC. The equivalent in IBM HTTP Server powered by Apache is the User directive, which is not configured by default. The User directive often is used on distributed systems where the parent Apache process is started as the root user ID. On z/OS, there is no need to use a “root” style user ID; therefore, it is not necessary to use this directive.

The LDAP authentication directives are different in IBM HTTP Server powered by Domino and IBM HTTP Server powered by Apache. Example 5-4 shows an LDAP configuration for IBM HTTP Server powered by Domino.

Example 5-4 IBM HTTP Server powered by Domino LDAP authentication directives

```
LDAPInfo PrimaryLdapServer {  
Host LDAPhostname  
Transport TCP  
ClientAuthType Basic  
ServerAuthType Basic  
ServerDN "cn=dgw, o=IBM, c=R0"  
ServerPasswordStashFile "StashFileName"  
UserSearchBase "o=IBM c=R0"  
GroupSearchBase "o=IBM c=R0"  
Referrals On  
}
```

The equivalent configuration for IBM HTTP Server powered by Apache is shown in Example 5-5.

Example 5-5 IBM HTTP Server powered by Apache LDAP authentication directives

```
<Location /secure>  
Order deny,allow  
Allow from all  
AuthName LDAPtx9name  
AuthBasicProvider ldap  
AuthType Basic  
AuthLDAPURL  
"ldap://ldap.example.com:1389/profiletype=user,sysplex=tx?rafid?sub?none"  
Require valid-user  
AuthLDAPBindDN "racfid=my-id,profiletype=user,sysplex=tx"  
AuthLDAPBindPassword my-password  
LdapReferrals on  
</Location>
```

For an IBM HTTP Server powered by the Domino Transport, Host, Port, UserSearchBase, and UserNameFilter directives, these directives are converted to the IBM HTTP Server powered by Apache *AuthLDAPURL* directive.

Note: The IHS AuthLDAPURL directive is part of the mod_auth_ldap module. The following syntax is used:

```
ldap://host:port/basedn?attribute?scope?filter
```

This string features the following elements:

- ▶ **host:port**
The name and port of the ldap server (defaults to localhost:389 for ldap, and localhost:636 for ldaps). To specify multiple redundant LDAP servers, you must list all servers separated by spaces.
- ▶ **basedn**
The DN of the branch of the directory from where all searches should start. It can be the top of your directory tree or a subtree in the directory.
- ▶ **attribute**
The attribute to search for. Although RFC 2255 allows a comma-separated list of attributes, only the first attribute is used, no matter how many are provided. If no attributes are provided, the default is to use uid. The attribute must be unique across all entries in the subtree you are using.
- ▶ **scope**
The scope of the search. It can be one or sub. A scope of base is also supported by RFC 2255, but is not supported by this module. If the scope is not provided or if base scope is specified, the default is to use a scope of sub.
- ▶ **filter**
A valid LDAP search filter. If not provided, this value defaults to (objectClass=*), which searches for all objects in the tree. Filters are limited to approximately 8000 characters (the definition of MAX_STRING_LEN in the Apache source code).

For more information about security aspects, see Chapter 7, “Security” on page 117.

5.2.7 Logging

The IBM HTTP Server powered by Domino AccessLog, AgentLog, RefererLog, ErrorLog, CgiErrorLog, ProxyAccessLog, and CacheAccessLog directives are replaced in IBM HTTP Server powered by Apache with a more powerful and flexible set of log directives called ErrorLog and CustomLog. You can use these directives for a virtual host container or for the whole server. For more about these directives, see 4.5.2, “Virtual hosting” on page 73.

The IBM HTTP Server powered by Domino log directives that are shown in Example 5-6 on page 92 can be replaced with the IBM HTTP Server powered by Apache log directives that are shown in Example 5-7 on page 92.

Example 5-6 IBM HTTP Server powered by Domino log directives

```
AccessLog /ihsconfig/dws/ihsdm001/logs/httpd-log
AgentLog /ihsconfig/dws/ihsdm001/logs/agent-log
RefererLog /ihsconfig/dws/ihsdm001/logs/referer-log
ErrorLog /ihsconfig/dws/ihsdm001/logs/httpd-errors
CgiErrorLog /ihsconfig/dws/ihsdm001/logs/cgi-error
ProxyAccessLog /ihsconfig/dws/ihsdm001/logs/httpd-proxy
CacheAccessLog /ihsconfig/dws/ihsdm001/logs/httpd-cache
```

Example 5-7 IBM HTTP Server powered by Apache log directives

```
CustomLog /ihsconfig/ihs/ihsam001/access_log common
CustomLog /ihsconfig/ihs/ihsam001/referer_log referer
CustomLog /ihsconfig/ihs/ihsam001/agent_log agent
ErrorLog /ihsconfig/ihs/ihsam001/error_log
```

For more information about how you can tailor the information that is recorded in these logs, see this website:

http://httpd.apache.org/docs/2.2/mod/mod_log_config.html#customlog

5.2.8 URL and file mapping directives

All URLs and file directives in IBM HTTP Server powered by Domino can be converted to corresponding URLs and file directives in IBM HTTP Server powered by Apache. The equivalent directives in IBM HTTP Server powered by Domino and IBM HTTP Server powered by Apache are listed in Table 5-1.

Table 5-1 URL and file directive comparison

IBM HTTP Server powered by Domino	IBM HTTP Server powered by Apache
<i>Pass</i>	<i>Alias</i>
<i>Exec</i>	<i>ScriptAlias</i>
<i>Map</i>	<i>Rewrite</i>
<i>Redirect</i>	<i>Redirect</i>
<i>Fail</i>	<i>Deny</i>
<i>Proxy</i>	<i>ProxyPass</i>

Migrating Pass directive

The IBM HTTP Server powered by Domino *Pass* directive can be converted to a simple IBM HTTP Server powered by Apache *Alias* directive. However, if you are using the third parameter (which is the IP address or site name), you must organize around containers, such as VirtualHost, Directory, and Location.

A sample set of directives from IBM HTTP Server powered by Domino is shown in Example 5-8 on page 93.

Example 5-8 HTTP Server Powered by Domino directives

```
Pass      /itsoInfo/* /ihsconfig/dws/ihsde001/itsoSecret/* w3.sc55.itso.ibm.com
Pass      /itsoInfo/* /ihsconfig/dws/ihsde001/itsoPublic/* wtsc55.itso.ibm.com
```

How these directives were converted to corresponding directives in IBM HTTP Server powered by Apache is shown in Example 5-9.

Example 5-9 HTTP Server Powered by Apache Alias directives

```
<VirtualHost wtsc55.itso.ibm.com:8230>
Alias /itsoInfo/ /ihsconfig/dws/ihsde001/itsoPublic/
</VirtualHost>
<VirtualHost w3.sc55.itso.ibm.com:8230>
Alias /itsoInfo/ /ihsconfig/dws/ihsde001/itsoSecret/
</VirtualHost>
```

Example 5-10 shows the directory structure that was used in association with the directives that are shown in Example 5-8.

Example 5-10 Directory layout used with IBM HTTP Server powered by Domino

```
EDMCAR @ SC55:/ihsconfig/dws/ihsde001>ls -lRtR itsoPublic itsoSecret
```

```
itsoSecret:
total 16
-rwxrwxr-x  1 EDMCAR  IHSRB13      66 Jul 28 20:24 main.html
```

```
itsoPublic:
total 16
-rwxrwxr-x  1 EDMCAR  IHSRB13      62 Jul 28 20:35 main.html
```

Example 5-11 shows the directory structure that was used in association with the directives in Example 5-9.

Example 5-11 Directory layout used with IBM HTTP Server powered by Apache

```
EDMCAR @ SC55:/ihsconfig/ihs/ihsae001/htdocs>ls -lRtR itsoPublic itsoSecret
```

```
./itsoPublic:
total 16
-rwxrwxr-x  1 EDMCAR  IHSRB13      62 Jul 28 20:39 main.html
```

```
./itsoSecret:
total 16
-rwxrwxr-x  1 EDMCAR  IHSRB13      66 Jul 28 20:39 main.html
```

Migrating Exec and Redirect directives

The IBM HTTP Server powered by Domino Exec directive can be converted to the IBM HTTP Server powered by Apache ScriptAlias directive and the Redirect is the same for both web servers, with the exception that wildcards are not used in IBM HTTP Server powered by Apache, as shown in Example 5-12 on page 94.

Example 5-12 Exec directive conversion to ScriptAlias directive

```
#IBM HTTP Server powered by Domino example:
Exec /cgi-bin/* /ihsconfig/dgw/ihsdm001/cgi-bin/*
Redirect      /oldcontext/*      http://servername/newpath/*

#IBM HTTP Server powered by Apache example:
ScriptAlias /cgi-bin/ "/ihsconfig/ihs/ihsam001/cgi-bin/"
Redirect permanent /oldcontext http://servername/newpath
```

The IBM HTTP Server powered by Apache Redirect directive can be used in the following instances:

- ▶ Redirect directive: Sends an external redirect that prompts the client to fetch a different URL.
- ▶ RedirectMatch directive: Sends an external redirect that is based on a regular expression match of the current URL.
- ▶ RedirectPermanent directive: Sends an external permanent redirect that prompts the client to fetch a different URL.
- ▶ RedirectTemp directive: Sends an external temporary redirect that prompts the client to fetch a different URL.

The DGW Map directive that is closest to IBM HTTP Server powered by Apache is the Rewrite directive. This directive is included in the mod_rewrite module. For more information about this directive, see 7.5, “Controlling access by using mod_rewrite” on page 128.

5.2.9 WebSphere Application Server plug-in

If you have servers that are using the WebSphere Application Server plug-in, see Chapter 9, “Plug-in for WebSphere Application Server” on page 137.

5.2.10 Timeouts

The timeout directives in IBM HTTP Server powered by Domino differ from the IBM HTTP Server powered by Apache timeout directives. IBM HTTP Server powered by Domino uses the InputTimeout, OutputTimeout, and ScriptTimeout directives that in IBM HTTP Server powered by Apache are translated into the Timeout directive. This directive specifies a number (in seconds) for the amount of time the server waits for certain events before failing a request. This directive can be specified within the server context and, if needed, within the virtual host context.

IBM HTTP Server powered by Domino PersistTimeout directive translates into the IBM HTTP Server powered by Apache KeepAliveTimeout directive, which expresses the amount of time (in seconds) the server waits for requests on a persistent connection. IBM HTTP Server powered by Apache also uses the KeepAlive, MaxKeepAliveRequests, and KeepAliveTimeout directives, as shown in Example 5-13.

Example 5-13 Timeout settings in IBM HTTP Server powered by Apache httpd.conf file

```
Timeout 180
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 10
```

If set to On, the KeepAlive directive enables HTTP persistent connections. The MaxKeepAliveRequests directive specifies the number of requests that are allowed on a persistent connection. The KeepAliveTimeout directive specifies the amount of time (in seconds) the server waits for requests on a persistent connection.

RequestReadTimeout directive

The mod_reqtimeout module provides the RequestReadTimeout directive, which can be used to control how long the server waits for content from the client. For more information, see this website:

https://publib.boulder.ibm.com/httperv/manual70/mod/mod_reqtimeout.html

5.2.11 Caching

For more information about caching features migration considerations, see Chapter 10, “Cache configuration” on page 145.

5.2.12 ASCII/EBCDIC considerations

The HTTP protocol stipulates that all request headers and response headers are transmitted in 8-bit ASCII. POST content and response content often are in ASCII, if it is text. Most programs that are running in IBM HTTP Server powered by Domino and in IBM HTTP Server powered by Apache examine and create the headers in EBCDIC, and the server translates them. The programs usually generate text content in EBCDIC. IBM HTTP Server powered by Apache translates Request POST content to EBCDIC if it is text. To have the IBM HTTP Server powered by Apache convert files in EBCDIC to ASCII, use the directives that are similar to the directives that are shown in Example 5-14.

Example 5-14 IBM HTTP Server powered by Apache directives

```
LoadModule charset_lite_module modules/mod_charset_lite.so
<IfModule mod_charset_lite.c>
  <Location / >
    CharsetSourceEnc IBM-1047
    CharsetDefault  ISO8859-1
  </Location>
</IfModule>
```

IBM 1047 is an EBCDIC character set, and ISO8859-1 is an ASCII set. If documents are stored in ASCII in the z/OS zFS, inform IBM HTTP Server powered by Apache by adding directives similar to these just before the closing the </IfModule> in Example 5-14. The directives must be similar to what is shown in Example 5-15.

Example 5-15 IBM HTTP Server powered by Apache directives for documents stored in ASCII

```
<Location /ascii_text/ >
  CharsetSourceEnc ISO8859-1
  CharsetDefault  ISO8859-1
</Location>
Alias /ascii_text/ "/usr/lpp/internet/ascii/"
```

The AddType directive in IBM HTTP Server powered by Apache is similar to that in IBM HTTP Server powered by Domino, but its options are in a different order. Also, IBM HTTP Server powered by Apache has no options for encoding (that is, ASCII/EBCDIC/binary) or for quality ratings.

In addition, IBM HTTP Server powered by Domino allowed for a CGI to write a header `Content-Encoding: ascii` or `Content-Encoding: binary` to control translation of response content from EBCDIC to ASCII. IBM HTTP Server powered by Apache ignores this response header.

Therefore, if it is impractical to separate the ASCII files from the EBCDIC files in the zFS, and they are identifiable by the file name extension, you might consider the use of a `LocationMatch` directive. For example, if your files that include the `.ascii` extension are HTML files in ASCII and your files that include the `.asctext` extension are plain text files in ASCII, you might consider directives that are shown in Example 5-16.

Example 5-16 Using LocationMatch directive

```
AddType text/html .ascii
AddType text/plain .asctext
<LocationMatch "\.(ascii|asctext)$" >
    CharsetSourceEnc ISO8859-1
    CharsetDefault   ISO8859-1
</LocationMatch>
```

Regarding SSI files (server-side includes, often named with the suffix `.shtml`), IBM HTTP Server powered by Domino handles these files even if they are stored on disk in ASCII. IBM HTTP Server powered by Apache requires that these files are stored in EBCDIC.

For a CGI that emits its response content in ASCII, use directives similar to the directives that are shown in Example 5-17.

Example 5-17 Directives for a CGI that emits response content in ASCII

```
<Location /ascii_exec/ >
    CharsetSourceEnc ISO8859-1
    CharsetDefault   ISO8859-1
</Location>
ScriptAlias /ascii_exec/ "/usr/lpp/internet/ascii_e/"
```

If this CGI is a z/OS shell script that emits ASCII content, it should be stored in EBCDIC for the command interpreter. It should still write its headers in EBCDIC, then write its response content in ASCII.

The input content side of the request uses the same rules. That is, if the CGI is governed by this ASCII Location container, IBM HTTP Server powered by Apache does *not* translate Request POST content to EBCDIC.

IBM HTTP Server powered by Domino features several special options for GWAPI programs that allow the program to control these ASCII/EBCDIC options. IBM HTTP Server powered by Apache cannot use these programs or these options. The following directives in IBM HTTP Server powered by Domino have no counterpart in IBM HTTP Server powered by Apache:

- ▶ `PostDataConv`
- ▶ `DetectUTF8 ON`
- ▶ `ENUExecs`
- ▶ `AddEncoding`

These directives in IBM HTTP Server powered by Domino have the following approximate counterparts in IBM HTTP Server powered by Apache:

- ▶ DefaultFsCp - CharSetSourceEnc
- ▶ DefaultNetCp - CharSetDefault
- ▶ AddLanguage - AddLanguage
- ▶ AddCharSet - AddCharSet
- ▶ AddType - AddType

The special case for nph- output is the same in IBM HTTP Server powered by Domino and IBM HTTP Server powered by Apache.

5.2.13 GWAPI

If GWAPI modules are used, you should determine their functionality and then determine whether the same functionality can be provided by a capability of IBM HTTP Server powered by Apache. If no corresponding capability is available, you might need to develop your own custom module. For more information about using modules in IBM HTTP Server powered by Apache, see Chapter 11, “Modules” on page 153.

5.2.14 Reverse Proxy

If DGW is used as a reverse proxy, IBM HTTP Server powered by Apache also can be configured for use as a reverse proxy. For more information, see this website:

<https://ibm.biz/BdrUa2>

5.2.15 Comparing DGW and IHS use of directives

For more information about the configuration directives difference between DGW and IHS and how the same logical function is achieved by the two different HTTP Servers, see this website:

<http://publib.boulder.ibm.com/httperv/ihsdiag/conversion.html>

5.2.16 Cleaning up PARMLIB

You should remove IMWHTTPD from any SCHED members of PARMLIBs.

5.3 Migrating Library Server

IBM Library Server provides a way to view IBM manuals by using a web browser. For more information, see this website:

<https://ibm.biz/BdrUay>

You can set up an IBM HTTP Server powered by Domino as the server that supports Library Server.

This section describes how you set up an IBM HTTP Server powered by Apache to support Library Server.

5.3.1 Set up in DGW

We set up an IBM HTTP Server powered by Domino and added the directives that are shown in Example 5-18. We also tested that the Library Server administration home page can be accessed.

Example 5-18 Directives in DGW to support Library Server

```
#-----  
#      BookManager BookServer  
#-----  
#  
Exec  /bookmgr-cgi/bookmgr.cmd*  /usr/lpp/booksrv/cgi-bin/EPHB00KS*  
Exec  /bookmgr-cgi/bookmgr.exe*  /usr/lpp/booksrv/cgi-bin/EPHB00KS*  
Exec  /bookmgr-cgi/*             /usr/lpp/booksrv/cgi-bin/*  
#  
Pass  /bookmgr/pictures/* /usr/lpp/booksrv/public/bookmgr/pictures/*  
Pass  /bookmgr/frames/*    /usr/lpp/booksrv/public/bookmgr/frames/*  
Pass  /bookmgr/*           /usr/lpp/booksrv/public/bookmgr/*
```

5.3.2 Set up in V8.5.5

In our IBM HTTP Server powered by Apache, we added the directives that are shown in Example 5-19.

Example 5-19 Directives added to IBM HTTP Server powered by Apache

```
Alias  /bookmgr/pictures /usr/lpp/booksrv/public/bookmgr/pictures  
Alias  /bookmgr/frames   /usr/lpp/booksrv/public/bookmgr/frames  
Alias  /bookmgr          /usr/lpp/booksrv/public/bookmgr  
  
ScriptAlias /bookmgr-cgi/bookmgr.cmd "/usr/lpp/booksrv/cgi-bin/EPHB00KS"  
ScriptAlias /bookmgr-cgi/bookmgr.exe "/usr/lpp/booksrv/cgi-bin/EPHB00KS"  
ScriptAlias /bookmgr-cgi/           "/usr/lpp/booksrv/cgi-bin/"
```

The Alias directives that are shown in Example 5-19 were added after the Alias directives in the httpd.conf file.

The ScriptAlias directives that are shown Example 5-19 were added after the ScriptAlias directives in the httpd.conf file.

Handling css files

The file in /usr/lpp/booksrv/public/bookmgr/lssstyles.css is a text file and is stored as ASCII in the UNIX System Services environment on z/OS. When accessed through Domino, it is downloaded as a binary file. It features ASCII text in the browser and is then used correctly.

In IBM HTTP Server powered by Apache, the default setup results in Apache trying to convert the file from ASCII to EBCDIC. The result is invalid characters in the browser and thus an invalid css file that the browser cannot use.

To resolve this issue, add the directive that is shown in Example 5-20 on page 99 near the bottom of the httpd.conf file.

Example 5-20 Directive to prevent character set conversion

```
SetEnvIf Request_URI /bookmgr/(lookat/index_files/.*\.css|.*\.css)$ no-xlate
```

The css files in the lookat/index_files directory are not used by Library Server. You can use the following simpler directive instead:

```
SetEnvIf Request_URI /bookmgr/.*\.css no-xlate
```

We show the slightly more complicated directive in Example 5-20 as an example of how to use a regular expression in the one directive to handle files in different subdirectories.

Environment Variable setup

The example that is shown in 5.3.2, "Set up in V8.5.5" on page 98 sets up a limited Library Server only. For a fully functional Library Server, you must specify various environment variables. How the /ihsconfig/ihs/ihsae65/bin/envvars file is updated to add directories that are associated with Library Server is shown in Example 5-21.

Example 5-21 Updating envvars

```
export JAVA_HOME="/usr/lpp/java_mounts/J7.0/J7.1"
export XERCESCROOT="/usr/lpp/ixm/IBM/xml4c-5_7"
export
PATH="/bin:./usr/sbin:/usr/lpp/internet/bin:/usr/lpp/internet/sbin:/usr/lpp/ldap/
bin:$JAVA_HOME/bin:$PATH"
LIBPATH=".:$XERCESCROOT/lib:/usr/lpp/internet/bin:/usr/lpp/internet/sbin:/usr/lpp/
ldap/lib:$JAVA_HOME/lib:$JAVA_HOME/bin:$LIBPATH"
export EPHConfigPath="/etc/booksrv/configs"
```

In the httpd.conf file, you also must add the directives that are shown in Example 5-22.

Example 5-22 Adding PassEnv directives

```
PassEnv JAVA_HOME
PassEnv XERCESCROOT
PassEnv LIBPATH
PassEnv PATH
PassEnv EPHConfigPath
```

CGI memory considerations

When we first attempted to access Library Server through our IBM HTTP Server powered by Apache, it did not succeed. In the error log, we found the following message:

```
CEE3512S An HFS load of module libicudata38.1.dll failed.
The system return code was 0000000157; the reason code was
0BDF019B.
```

This message indicated that there was a lack of memory.

To determine how much memory was available to our CGI programs, we found a program that is named jdkiv available at this website:

<http://www.ibm.com/support/docview.wss?uid=swg21252834>

We placed this file in a directory on our z/OS LPAR. Then, we created the shell program that is shown in Example 5-23.

Example 5-23 Shell program to display useful information

```
#!/bin/sh
printf "Content-Type: text/plain\r\n\r\n"
date
env
id
/u/edmcarg/jdkiv
ulimit -a
echo 'ulimit -A 2000000'
ulimit -A 2000000
echo 'ulimit -M 800'
ulimit -M 800
echo 'ulimit setting now...'
ulimit -a
```

We added the ulimit statements to see whether the available memory for CGI programs can be adjusted.

We stored this shell program in /ihsconfig/ihs/ihsae65/cgi-bin/showEnv.sh and started it by using the following URL:

<http://wtsc55.itso.ibm.com:8265/cgi-bin/showEnv.sh>

The browser output included the following line from the jdkiv program:

getrlimit reports RLIMIT_AS as current: 35717120, max:2147483647

This value of 35717120 is insufficient to run the Library Server CGI programs and thus was increased. This value is also referred to as the *soft limit*. The result of the ulimit commands we issued is shown in Example 5-24.

Example 5-24 Output from ulimit commands

```
core file          unlimited
cpu time           27011
data size          unlimited
file size          unlimited
stack size         unlimited
file descriptors   65535
address space      unlimited
memory above bar   512m
ulimit -A 2000000
ulimit -M 800
ulimit setting now...
core file          unlimited
cpu time           27011
data size          unlimited
file size          unlimited
stack size         unlimited
file descriptors   65535
address space      2000000k
memory above bar   512m
```

The output shows that the `ulimit -A` command increased the address space value, which is the soft limit.

An Apache directive that is named `RLimitMEM` can be used to increase the soft limit. However, during testing we found that this increase did not work. APAR PI31566 was opened to correct this issue.

ASSIZEMAX value in OMVS Segment

To resolve this memory issue, the `ASSIZEMAX` value in the OMVS segment of the User ID the server is running under can be adjusted. We used the following command:

```
ALTUSER EDMCAR OMVS(ASSIZEMAX(2147483647))
```

After restarting the server and restarting the shell, the `jdkiv` program generated the following message:

```
getrlimit reports RLIMIT_AS as current:2147483647, max:2147483647
```

5.3.3 Testing Library Server

We used the following URL to test whether we can access Library Server through IBM HTTP Server powered by Apache:

```
http://wtsc55.itso.ibm.com:8265/bookmgr-cgi/bookmgr.exe/ADMINISTRATION
```

This test succeeded with the browser showing the output that is shown in Figure 5-1.

Library Server Administration

Shelf, Bookcase and LibraryCenter Management

Create Shelf Create Bookcase Create LibraryCenter

Manage Shelves Manage Bookcases Manage Document Indexes

Manage InfoCenters

Administration password:

Save Settings Rebuild Catalog Clear Cache Delete All Notes Reset Java Bridge Status

Settings, Paths and Names

Enter directories for document collections: 1. /usr/lpp/booksrv/books

Enter descriptive document collection names: 1. Classic Novels

Figure 5-1 Library Server administration home page



Scalability and workload management

This chapter describes the different ways that IBM HTTP Server powered by Apache (IHS V8.5.5) and IBM HTTP Server powered by Domino (V5.3 or DGW) provide scalability support. It also describes how they use WLM.

This chapter includes the following topics:

- ▶ 6.1, “Overview” on page 104
- ▶ 6.2, “DGW approach” on page 104
- ▶ 6.3, “IHS V8.5.5 approach” on page 105
- ▶ 6.4, “V8.5.5 support for WLM” on page 110
- ▶ 6.5, “Working with WLM in IHS V8.5.5” on page 111
- ▶ 6.6, “Summary” on page 116

6.1 Overview

Scalability is an important issue for HTTP Servers because the way the HTTP Server can be configured to handle increases in activity can have an effect on the available resources.

Ideally, you want your HTTP Server to dynamically increase its capability to handle increases in activity. This ability typically involves starting more processes and the use of more memory. After the increase in activity subsides, you want the HTTP Server to dynamically reduce the resources it is using by stopping processes and freeing memory.

V8.5.5 and V5.3 (DGW) provide dynamic scalability support, which they provide in different ways.

6.2 DGW approach

The DGW approach is heavily integrated into the z/OS Workload Management (WLM) capability. The approach that it uses is referred to as the *Scalable Server mode*.

How the Scalable Server mode works is described briefly in this section to compare with the V8.5.5 approach. Figure 6-1¹ shows an overview of how the Scalable Server mode balances requests across multiple DGW servers.

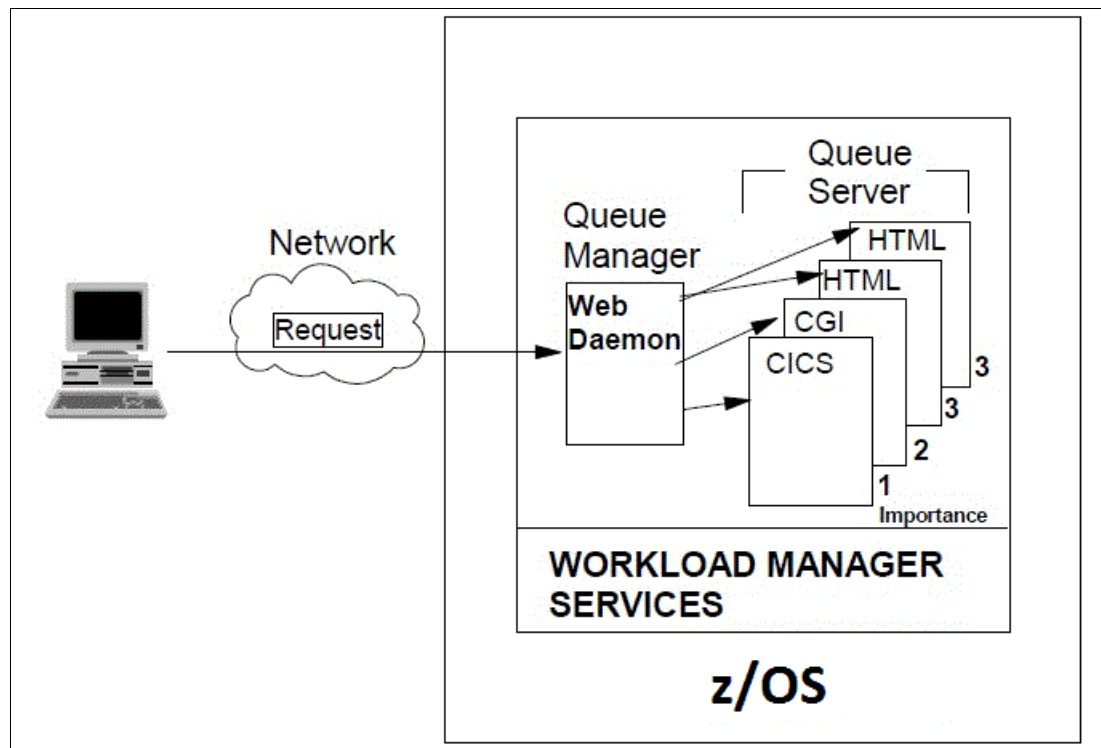


Figure 6-1 DGW Scalable Server mode overview

¹ For more information, see *Enterprise Web Serving with the Lotus Domino Go Webserver for OS/390*, SG24-2074.

The address spaces include the following functions:

- ▶ Queue Manager

Queue manager is the front-end process. It receives the request from the client. If an application environment (ApplEnv) definition is available (in the `httpd.conf` file and the WLM policy), queue manager routes the request to the appropriate queue server by putting the request onto the WLM queue. If no ApplEnv is defined, it handles the request. Only one queue manager per web server is used in any one z/OS system.

- ▶ Queue Server

Queue server is the process that runs the client's request (except for secured connection requests and requests that are not defined to run in a queue server). It searches its associated queue and picks up the requests to process. Multiple queue servers can be available for each queue and for any web server in one z/OS system.

WLM manages these processes based on the policies that are defined.

To meet the performance goals of the system, WLM controls the number of queue servers that run requests on connected queues. If so many requests are received that the current queue servers cannot meet the performance goals, WLM starts more queue servers. If the demand for servers is low, WLM stops some idle queue servers to reduce the allocated system resources.

For more information about Scalable Server mode and how to set it up, see this z/OS 1.13 IBM Knowledge Center website:

<https://ibm.biz/Bdr5rB>

6.3 IHS V8.5.5 approach

The V8.5.5 approach is simpler than the DGW approach and is not tied to WLM. For more information about integrating with WLM, see 6.4, “V8.5.5 support for WLM” on page 110 describes how V8.5.5.

6.3.1 Multi-processing module

V8.5.5 uses the Apache Multi-Processing Module (MPM) to provide scalability support. A server supports several MPMs. Versions of IBM HTTP Server powered by Apache up to and including V8.0 use the worker MPM. For more information, see this website:

<http://httpd.apache.org/docs/2.0/mod/worker.html>

IBM HTTP Server powered by Apache versions from V8.5 onward use the event MPM. For more information, see this website:

<http://httpd.apache.org/docs/current/mod/event.html>

The main concept of the worker and event MPM is to have available several child servers, each of which contains several threads. This approach aims to achieve a balance between system resources and performance.

The event MPM is based on the worker MPM but provides what is called *async I/O support*. In this support, threads are not tied to a TCP/IP connection continuously, which results in better performance and the ability to manage many more connections at once.

A single control process (which is referred to as *the parent*) starts the child processes.

The Apache terminology can be confusing initially because it refers to starting child processes, but then uses directives that include the word “server”, such as *StartServers*. These *StartServers* directives refer to the child processes.

MaxClients and ThreadsPerChild directives

Several directives can be used to control how the worker MPM behaves. The key following key directives are available:

- ▶ **MaxClients**

This directives specifies the maximum number of simultaneous client connections that are allowed to the server.

- ▶ **ThreadsPerChild**

This directives specifies how many threads are in each child process (server).

It is these directives that V8.5.5 uses to determine how many child processes can be started. For example, assume that the following set is in the configuration file:

```
MaxClients      100
ThreadsPerChild 5
```

If there are 100 connections to the server, $100/5 = 20$ child processes are active in V8.5.5.

ThreadLimit parameter

The value that is specified for the *ThreadLimit* directive is the maximum value to which the value of the *ThreadsPerChild* can be increased dynamically by using a **restart** command.

Reacting to changes in activity

V8.5.5 checks every second to determine how it is managing the current activity. If it detects that the number of available threads in the running child processes are not enough to handle the number of requests that are received, it dynamically starts another child process to provide more threads for new requests on which to be dispatched.

If the activity drops off and more threads are available than needed, it stops child processes to free up resources, such as memory.

Controlling available threads

When V8.5.5 receives new requests, there is a wait if there were no available threads while V8.5.5 creates a child process before the requests can be processed. Although this process does not take an inordinate amount of time, it is still better to avoid this situation.

The *MinSpareThreads* and *MaxSpareThreads* directives provide a way to control the number of idle threads that are available at any time.

V8.5.5 assesses every second how many idle threads there are available in all child processes. It stops or starts child processes so that the number of idle threads is within the values of *MinSpareThreads* and *MaxSpareThreads*. This check provides an initial buffer of available threads that can immediately start processing new requests and allow V8.5.5 on its next check of how it is handling the workload to dynamically increase the number of child processes, if needed.

For more information about tuning Apache, see this website:

http://publib.boulder.ibm.com/httperv/ihsdiag/ihs_performance.html

Although z/OS is not addressed, the concepts that are described on the website still apply.

6.3.2 How V8.5.5 looks on z/OS

For an example of what V8.5.5 looks like when it is running on z/OS, see Figure 6-2. This example shows the logical view of a V8.5.5 server running on z/OS and how it corresponds to the started tasks that are seen in SDSF.

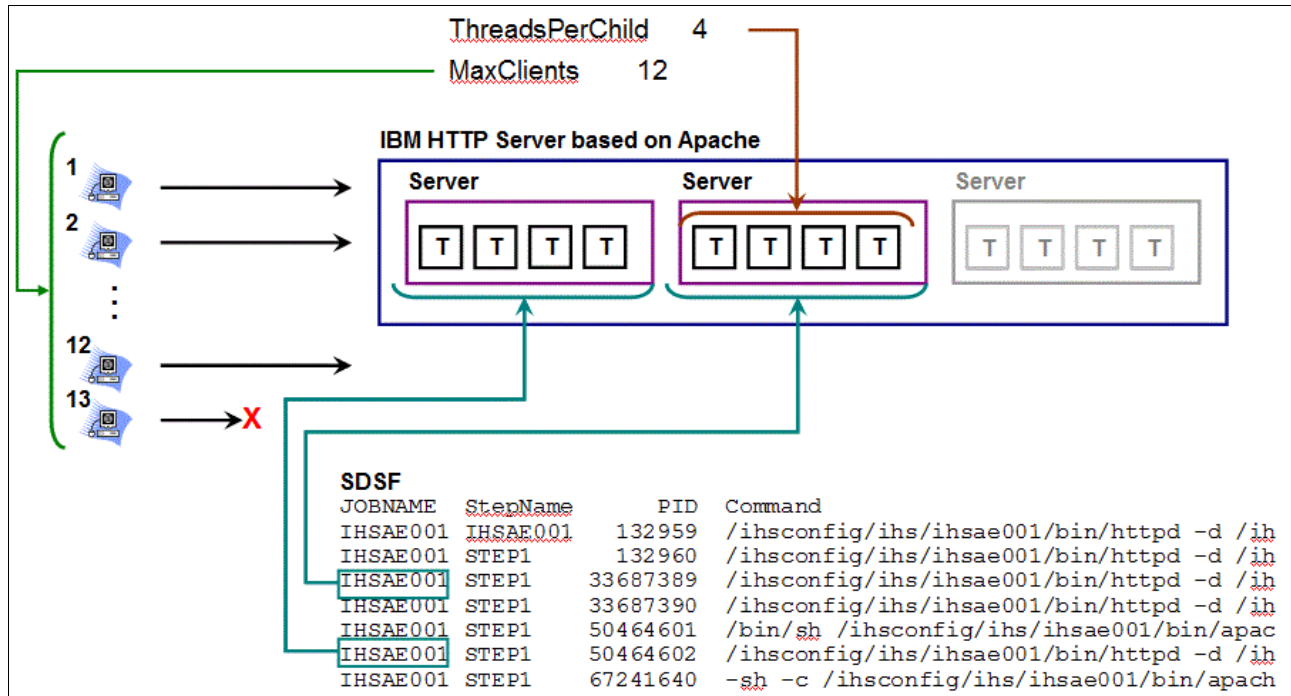


Figure 6-2 IBM HTTP Server powered by Apache on z/OS

In this example, we set the following directives:

- ▶ ThreadsPerChild: 4
- ▶ MaxClients: 12
- ▶ MinSpareThreads: 6
- ▶ MaxSpareThreads: 8

With these settings, a maximum of 12 simultaneous connections to V8.5.5 are available. Any other connections are rejected after the maximum is met.

Two active child processes (labeled as Server) are available. Each process features four threads because ThreadsPerChild is set to four.

If requests are received, V8.5.5 attempts to keep the number of available threads across all child processes between the values that are specified for MinSpareThreads and MaxSpareThreads. In this case, V8.5.5 has two child processes that are running that results in eight available threads because these values are set to six and eight.

V8.5.5 can be started as a started task or started from a user session that is logged on by using Telnet or in OMVS. Regardless of which method is used, you see a display in SDSF similar to what is shown in Figure 6-2. In this case, we started V8.5.5 as a started task that is named IHSAE001. If we started it from a user session, the value in the JOBNAME column shows as the user's TSO ID.

If 12 requests arrived, V8.5.5 starts the third child process, which is shown in the gray box in Figure 6-2. In SDSF, you see that one more IHSAE001 appears as a result.

6.3.3 Example of dynamic scalability

To demonstrate how V8.5.5 allows you to dynamically change its runtime configuration to support changes in workload, we set up our server with the following directives:

- ▶ ThreadLimit: 8
- ▶ MaxClients: 12
- ▶ ThreadsPerChild: 4

In this set up, the server can support 12 simultaneous connections at most and can start three child processes with four threads each. The ThreadLimit value of eight means that the number of threads per child process can be increased 4 - 8, as required.

We then used JMeter to simulate 20 simultaneous users that are trying to run the sleep servlet with a sleep time of 2 seconds. In JMeter, we saw that the response time was erratic and averaged 4 - 6 seconds. We then changed the directives to the following values:

- ▶ MaxClients: 24
- ▶ ThreadsPerChild: 8

Note: V8.5.5 allows you to change the MaxClients and ThreadsPerChild values by using a graceful restart. However, the ThreadLimit value cannot be changed unless a full stop and start of the server is completed.

We then issued the `s ihsae002,action='graceful'` command to perform a graceful restart of our server while the load test was still running.

This restart resulted in the server now having eight threads per child process and needing to start only three child processes. In JMeter, we immediately saw the effect of this change, as the average response time settled around the 2-second mark, as shown in Figure 6-3 on page 109.

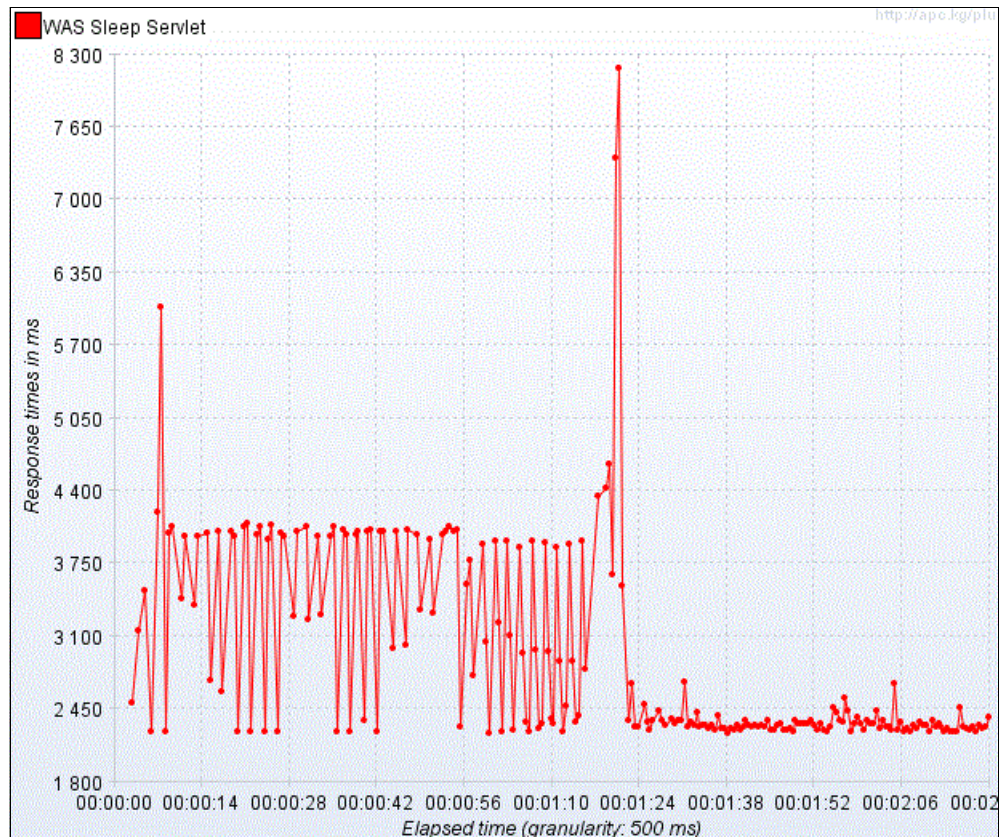


Figure 6-3 Result of dynamic reload to increase *ThreadsPerChild*

6.3.4 Sizing your server

In this section, we describe an approach to setting the parameters that we described to suit your environment to optimize use of resources.

MaxClients

Start by deciding on the maximum number of clients that you expect to connect at any time. For example, assume that the maximum number that you must support is 600. The directive is set as shown in the following example:

```
MaxClients 600
```

ThreadsPerChild

Next, determine the maximum number of threads that each server process can handle. This number can vary depending on your system capacity. Assume that a maximum value of 100 is reasonable, especially if SSL connections (which are computing-intensive) are used. In this example, we use 100 and code the directive as shown in the following example:

```
ThreadsPerChild 100
```

ServerLimit

The specified value for *ServerLimit* limits the maximum number of child processes that can be started. However, the maximum number of child processes that is started is equal to $\text{MaxClients} / \text{ThreadsPerChild}$ (in our case: $600 / 100 = 6$). No extra child processes are started if the result of $\text{MaxClients} / \text{ThreadsPerChild}$ is less than the value set for *ServerLimit*.

Setting `ServerLimit` to a value less than the result of `MaxClients/ThreadsPerChild` results in that lower number of child processes being started. In this case, the server does not have enough child processes and threads to handle the maximum number of clients. Therefore, this setting is not recommended.

MaxSpareThreads

`MaxSpareThreads` specifies the maximum number of idle threads that is maintained by the server in anticipation of new requests. If this setting is too low, delays might occur as new requests are queued. If the setting is too high, excess memory is used by the idle threads.

Reasonable values are any multiple of `ThreadsPerChild`, but significantly larger than `MinSpareThreads`. Specifying the same value as `MaxClients` prevents IHS from reclaiming idle threads. In this example, we set the value to 300.

MinSpareThreads

`MinSpareThreads` must be higher than the number of new requests that are received in 1 second. If the specified value is too low and V8.5.5 is running out of available threads, a delay of a few seconds can happen before the available threads are created.

Use a value that is approximately equal to 10% of `MaxClients`. In this example, we chose a value of 60 and coded the directive as shown in the following example:

```
MinSpareThreads 60
```

6.4 V8.5.5 support for WLM

IHS Version 8.5.5 introduces WLM support. This support allows requests that are received by V8.5.5 to be classified to a WLM service class.

The way to set up V8.5.5 to work with WLM is simpler than the DGW approach. It requires only the setup of classification definitions in WLM and the specification of the new WLM-related directives in the V8.5.5 `httpd.conf` file.

Enabling WLM support

To enable V8.5.5 for WLM support, the module that provides this support must be added to the `httpd.conf` file. We added the required directive after the `LoadModule` directives, as shown in the following example:

```
#LoadModule rewrite_module modules/mod_rewrite.so
#LoadModule deflate_module modules/mod_deflate.so
# WLM
LoadModule wlm_module modules/mod_wlm.so
```

WLM directives

The following new WLM directives are available:

- ▶ `wlmSubSysType`: Specifies a `SubSystemType` value that is defined in WLM.
- ▶ `wlmTranClass`: Specifies a value that is defined under the `Name` heading in the `Qualifier` part of the WLM ISPF windows (with the `Type` field set to a value of `TC`).
- ▶ `wlmCollectionName`: Specifies a value that is defined under the `Name` heading in the `Qualifier` part of the WLM ISPF windows (with the `Type` field set to a value of `CN`).

For more information about these directives, see this website:

http://publib.boulder.ibm.com/htpserv/manual70/mod/mod_wlm.html

6.5 Working with WLM in IHS V8.5.5

V8.5.5 is based on Apache and provides great flexibility in how you code the directives in the `httpd.conf` file. This paper cannot cover every possible configuration option of how you can use the WLM directives in V8.5.5. Instead, it provides one example that shows how we used these directives to classify different requests.

6.5.1 Mapping app requests to one WLM transaction class as default approach

The simplest (and in essence the default) approach is to map all requests to one WLM transaction class. To complete this mapping, we added the following lines at the bottom of our V8.5.5 `httpd.conf` file:

```
# Default WLM Transaction Class
wlmSubSysType CB
wlmCollectionName IHSE01
wlmTranClass IHSEWLM1
```

If we added no other directives, all requests that are received by V8.5.5 are classified to the IHSEWLM1 transaction class.

6.5.2 Mapping an application for a specific virtual host

We then wanted to classify requests for a specific application for a specific virtual host to be mapped to a specific transaction class. In our example, we wanted requests that start with IBMTTools for the `wtsc55.itso.ibm.com` host to be classified to the IHSEWLM2 transaction class. We achieved this mapping by adding the following directives near the bottom of the `httpd.conf` file:

```
<VirtualHost wtsc55.itso.ibm.com:8235>
<Location /IBMTTools/* >
wlmTranClass IHSEWLM2
</Location>
</VirtualHost>
```

Any requests that are received for the `wtsc55.itso.ibm.com` host that did not start with IBMTTools are classified to the IHSEWLM1 transaction class because it is the default class.

6.5.3 Mapping multiple applications within a specific virtual host

Next, we wanted different applications for a specific virtual host to be mapped to specific transaction classes. In our example, we wanted requests that are sent to the `w3.sc55.itso.ibm.com` virtual host to be classified as shown in the following examples:

- ▶ Requests starting with `ItsoTools` are classified to the IHSEWLM3 transaction class
- ▶ Requests starting with anything else are classified to the IHSEWLM4 transaction class

We added the following directives:

```
<VirtualHost w3.sc55.itso.ibm.com:8235>
<LocationMatch "/ItsoTools/*">
wlmTranClass IHSEWLM3
</LocationMatch>
wlmTranClass IHSEWLM4
</VirtualHost>
```

6.5.4 Connecting WLM directives and WLM setup

Now that the WLM directives are coded, we set up corresponding definitions in the WLM ISPF panels. Figure 6-4 shows how the WLM directives that we specified correspond to values that are specified in the WLM ISPF panels.

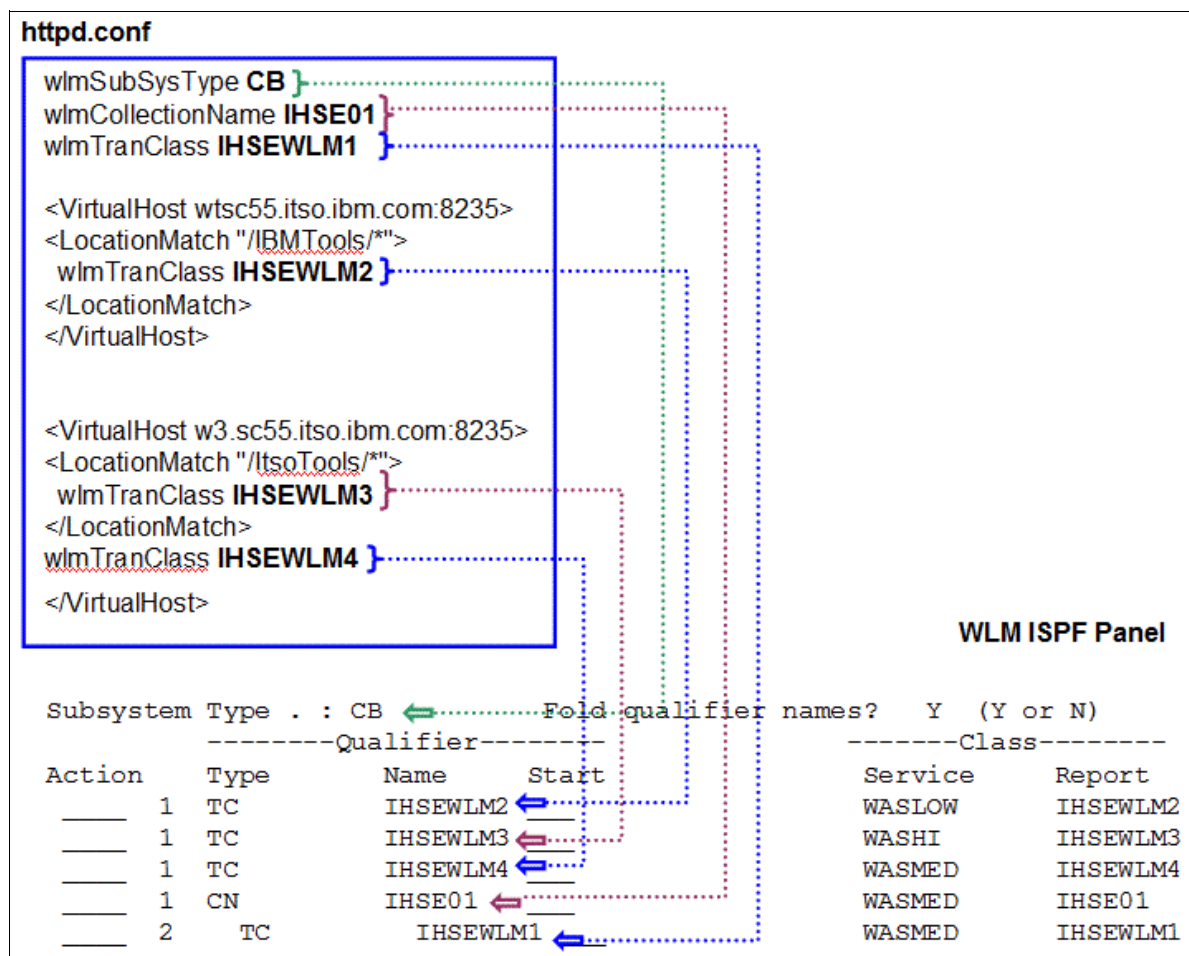


Figure 6-4 Mapping WLM directives to WLM ISPF panels settings

We mapped the different transaction classes to different WLM service classes.

The DGW scalable approach is that each queue server can process requests only for one WLM transaction classification. V8.5.5 allows requests with different WLM classifications to run within the same child process with each child process being a single started task.

6.5.5 WLM in action

To demonstrate the WLM classification in V8.5.5, we used JMeter to send requests to our server. JMeter is an open source utility that is available from Apache that is used to run load tests.

For more information, see this website:

<http://jmeter.apache.org/>

We set up the following JMeter load test sessions on our Windows workstation:

- ▶ Two users that are sending requests to wtsc55.itso.ibm.com to run requests that start with IBMTools.
- ▶ Three users that are sending requests to wtsc55.itso.ibm.com to run that start with ItsoTools.
- ▶ Two users that are sending requests to wtsc55.itso.ibm.com to run that start with IBMTools.
- ▶ Three users that are sending requests to wtsc55.itso.ibm.com to run that start with ItsoTools.

We then started these load tests. The requests that are received by V8.5.5 are routed to a WebSphere Application Server on z/OS through the WebSphere Application Server plug-in that is configured in the HTTP Server. The requests all start the same servlet, which enters a wait state for 2 seconds. This approach was used so that requests run for a comparatively long time to allow them to show up in our various displays.

What we saw in SDSF

In SDSF, we entered the **ENC** command to show active enclaves that are running in the z/OS LPAR. This result is the WLM classification of the requests because the requests are managed as enclaves by z/OS.

In SDSF we saw the following output:

```
SDSF ENCLAVE DISPLAY SC55      ALL                      LINE 1-2
COMMAND INPUT ==>                                     S
NP   NAME                      SStype Status   SrvClass Per PGN RptClass
    380011E8A4                  CB    ACTIVE   WASHI      1   IHSEWLM3
    540011E8B0                  CB    ACTIVE   WASHI      1   IHSEWLM3
    640011E8AE                  CB    ACTIVE   WASMED     1   IHSEWLM4
    880011E8A8                  CB    ACTIVE   WASMED     1   IHSEWLM1
    8C0011E8A6                  CB    ACTIVE   WASLOW     1   IHSEWLM2
    A40011E8AA                  CB    ACTIVE   WASLOW     1   IHSEWLM2
    AC0011E8AC                  CB    ACTIVE   WASMED     1   IHSEWLM1
```

This output shows that the different requests are being classified to the appropriate WLM service and report classes as expected.

View from RMF

In IBM RMF, you can see the WLM view of how requests were being processed, as shown in Example 6-1.

Example 6-1 Using RMF to view of how requests are processed

RMF V1R13 Sysplex Summary - WTSCPLX1					Line 16 of 29					
Command ==>					Scroll ==> CSR					
WLM Samples: 400					Systems: 16 Date: 06/21/13 Time: 21.48.20 Range: 100					Sec
>>>>>>>XXXXXXXXXXXXXXXXXXXX<<<<<<<<										
Service Definition: WPS					Installed at: 06/21/13, 20.30.35					
Active Policy: ALLCENT					Activated at: 06/21/13, 20.30.44					
----- Goals versus Actuals -----										
		Exec Vel		--- Response Time ---		Perf	Trans	--Avg. Resp. Time--		Time-
		Goal Act		---Goal--- --Actual--		Indx	Ended	WAIT	EXECUT	ACTUAL
Name	T	I	Goal	Act			Rate	Time	Time	Time
WASHI	S	1	0.0	0.200	90%	84%	****	9.450	0.000	0.316 0.317
WASLOW	S	3	0.0	5.000	90%	100%	0.50	0.270	0.004	2.011 2.015
WASMED	S	2	0.0	1.000	90%	11%	4.00	0.650	0.006	1.793 1.799
DDF	R		0.0					0.730	0.000	0.001 0.001
IHSEWLM1	R		0.0					0.350	0.006	1.606 1.612
IHSEWLM2	R		0.0					0.270	0.004	2.011 2.015
IHSEWLM3	R		0.0					0.320	0.007	2.009 2.016
IHSEWLM4	R		0.0					0.300	0.007	2.010 2.017

This example is contrived because we are running requests that enter a wait state for 2 seconds. What it does show is how WLM is measuring how well the z/OS environment is meeting the specified goals for our requests. Consider the following points:

- ▶ The requests that are running in service class WASHI are not meeting the goal of 90% of requests that are completing in less than 0.2 seconds.
- ▶ The requests that are running in WASMED are also not meeting the goal of 90% of requests that are completing in less than 1 second because the response time is 1.793 seconds.
- ▶ The requests that are running in WASLO are achieving the goal of 90% of requests completing in less than 5 seconds because the average response time is 2.011 seconds.

RMF also shows information about the transaction classes.

V8.5.5 server-status output

V8.5.5 provides a module that is named `mod_status` with which you can send a request to the server to get information about the requests being processed. During the load test, we sent the following request to obtain the status information:

`http://wtsc55.itso.ibm.com:8235/server-status`

The information that was received is shown in Figure 6-5.

M	CPU	SS	Req	Conn	Child	Slot	Client	VHost	Request
W	0.02	0	0	0.0	0.09	0.13	9.190.237.92	wtsc55.itso.ibm.com	GET /ItsoTools/Sleeper?SleepAmount=2 HTTP/1.1
W	0.02	0	0	0.0	0.07	0.11	9.190.237.92	wtsc55.itso.ibm.com	GET /ItsoTools/Sleeper?SleepAmount=2 HTTP/1.1
W	0.02	1	0	0.0	0.10	0.13	9.190.237.92	w3.sc55.itso.ibm.com	GET /ItsoTools/Sleeper?SleepAmount=2 HTTP/1.1
W	0.02	0	0	0.0	0.08	0.11	9.190.237.92	wtsc55.itso.ibm.com	GET /IBMTTools/Sleeper?SleepAmount=2 HTTP/1.1
_	0.00	1	52	0.0	0.01	0.02	9.190.237.92	wtsc55.itso.ibm.com	GET /server-status HTTP/1.1
W	0.00	0	0	0.0	0.01	0.02	9.190.237.92	wtsc55.itso.ibm.com	GET /server-status HTTP/1.1
_	0.00	8	2007	0.0	0.00	0.01	9.190.237.92	wtsc55.itso.ibm.com	GET /IBMTTools/Sleeper?SleepAmount=2 HTTP/1.1
_	0.00	4	49	0.0	0.01	0.02	9.190.237.92	wtsc55.itso.ibm.com	GET /server-status HTTP/1.1
W	0.09	0	0	0.0	0.05	0.07	9.190.237.92	wtsc55.itso.ibm.com	GET /IBMTTools/Sleeper?SleepAmount=2 HTTP/1.1
W	0.09	1	0	0.0	0.05	0.08	9.190.237.92	w3.sc55.itso.ibm.com	GET /ItsoTools/Sleeper?SleepAmount=2 HTTP/1.1
W	0.09	0	0	0.0	0.06	0.08	9.190.237.92	w3.sc55.itso.ibm.com	GET /IBMTTools/Sleeper?SleepAmount=2 HTTP/1.1
W	0.09	0	0	0.0	0.06	0.08	9.190.237.92	w3.sc55.itso.ibm.com	GET /IBMTTools/Sleeper?SleepAmount=2 HTTP/1.1

Figure 6-5 V8.5.5 server status information

Information from RMF WLM report

Because WLM information is recorded in SMF records, we ran a batch job to produce an RMF WLM report for the period of our load test. An extract from this report (which shows information about the IHSEWLM4 report class) is shown in Example 6-2.

Example 6-2 Displaying the IHSEWLM4 report class

REPORT BY: POLICY=ALLCENT				REPORT CLASS=IHSEWLM4			
				DESCRIPTION =IHS ITS0			
-TRANSACTIONS-	TRANS-TIME	HHH.MM.SS.TTT	--DASD I/O--	---	SERVICE---		
AVG	0.47	ACTUAL	2.013	SSCHRT	0.0	IOC	0
MPL	0.47	EXECUTION	2.007	RESP	0.0	CPU	2735
ENDED	140	QUEUED	5	CONN	0.0	MSO	0
END/S	0.23	R/S AFFIN	0	DISC	0.0	SRB	0
#SWAPS	0	INELIGIBLE	0	Q+PEND	0.0	TOT	2735
EXCTD	0	CONVERSION	0	IOSQ	0.0	/SEC	5
AVG ENC	0.47	STD DEV	0				
REM ENC	0.00					ABSRPTN	10
MS ENC	0.00					TRX SERV	10

The report for the SMF interval shows the following useful information:

- ENDED: Number of requests that completed
- END/S: Transaction rate
- ACTUAL: Average response time

Not shown is the information about how much CPU was used. Although in our case this amount was negligible, the amount from a real workload provides accurate information about how much CPU was used.

6.6 Summary

V8.5.5 and DGW provide scalability support, but in different ways. Both also use WLM. V8.5.5 provides a simpler but effective and robust way of supporting any of your scalability requirements.

Additionally, its support for WLM allows you to classify requests so that z/OS can prioritize workloads when the system is under heavy load.



Security

This chapter describes the use of security with the IBM HTTP Server powered by Apache on z/OS and includes the following topics:

- ▶ 7.1, “Security overview” on page 118
- ▶ 7.2, “Configuring V8.5.5 for your security requirements” on page 118
- ▶ 7.3, “SSL and Session ID” on page 122
- ▶ 7.4, “Configuring SSL support” on page 123
- ▶ 7.5, “Controlling access by using mod_rewrite” on page 128
- ▶ 7.6, “Caching and security considerations” on page 129

7.1 Security overview

Security can be used with IBM HTTP Server powered by Apache on z/OS in the following ways:

- ▶ Thread level security
An independent security environment can be set for each thread that is running under HTTP Server, which means that every client that connects to the server has its own security environment.
- ▶ HTTPS/SSL support
HTTP Server fully supports the Secure Sockets Layer (SSL) protocol. HTTPS uses SSL as a sublayer under the regular HTTP layer to encrypt and decrypt HTTP requests and HTTP responses. By default, HTTPS uses port 443 for serving instead of HTTP port 80.
- ▶ LDAP support
The Lightweight Data Access Protocol (LDAP) specifies a simplified way to retrieve information from an X.500-compliant directory in an asynchronous, client/server type of protocol.
- ▶ Certificate authentication
As part of the SSL support, HTTP Server can use certificate authentication and act as a certificate authority.
- ▶ Proxy support
IBM HTTP Server powered by Apache can be set up as a proxy server.

For more information, see the following IBM Knowledge Center website:

<https://ibm.biz/Bdr5ZT>

7.2 Configuring V8.5.5 for your security requirements

You can customize security in V8.5.5 in many ways. For more information about the security capabilities of V8.5.5, see this website:

<https://ibm.biz/Bdr5Yd>

This website describes such topics as the use of certificates and authenticating with SAF.

Although it is beyond the scope of this paper to demonstrate all security capabilities, the following section describes security that controls access to a Rexx program in the `cgi-bin` directory. The Rexx program can access output in the JES2 Spool by using SDSF APIs.

The Rexx program that is described is available from the following IBM Techdoc:

<http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD106087>

The following sections are from this IBM Techdoc and describe four approaches to controlling access:

- ▶ Allowing unauthenticated access
- ▶ Allowing all authenticated user access
- ▶ Allowing authenticated user that belongs to a group access
- ▶ Allowing authenticated user access with client credentials

7.2.1 Allowing unauthenticated access

After you set up your own V8.5.5, the default level of security is used. This default level means that no authentication is required to start the Rexx program. The implication is that anyone in your organization who knows the URL can start it.

Although anyone can start the URL, they can view only the JES spool output that the user ID under which IBM HTTP Server is running is authorized to view. All access to SDSF and JES spool is done under the user ID of the HTTP Server, which must be limited to only what is required.

7.2.2 Allowing all authenticated user access

The next level of access control is to require that only users who supply a user ID and password that are validated by using the z/OS SAF interface can access the Rexx program. However, access to SDSF and JES spool remains under the user ID of the HTTP Server. This arrangement is often referred to as *client-as-server access*. To achieve this control, modify the V8.5.5 `httpd.conf` file as described in this section.

In the `httpd.conf` file that your V8.5.5 is using, ensure that the following directives are present:

- ▶ `LoadModule auth_basic_module modules/mod_auth_basic.so`
- ▶ `LoadModule authz_user_module modules/mod_authz_user.so`

In the same area of the `httpd.conf` file, add the following directives:

- ▶ `LoadModule authnz_saf_module modules/mod_authnz_saf.so`
- ▶ `LoadModule authz_default_module modules/mod_authz_default.so`

After adding these two lines, that area of the `httpd.conf` file resembles the file that is shown in Example 7-1.

Example 7-1 httpd.conf after adding two LoadModule directives

```
LoadModule authn_file_module modules/mod_authn_file.so
LoadModule authz_user_module modules/mod_authz_user.so
# IBM-Rexx Added next 2 line to support SAF authentication
LoadModule authnz_saf_module modules/mod_authnz_saf.so
LoadModule authz_default_module modules/mod_authz_default.so
#LoadModule authz_groupfile_module modules/mod_authz_groupfile.so
LoadModule include_module modules/mod_include.so
```

At the bottom of the `httpd.conf` file, add the lines that are shown in Example 7-2.

Example 7-2 Directives to set up security controls

```
<Location ~ "/(sdsfViewer.rx*)">
  AuthName zosSdsfViewer
  AuthType Basic
  AuthBasicProvider saf
  Require valid-user
  AuthSAFExpiration "EXPIRED! oldpw/newpw/newpw"
  AuthSAFReEnter "Enter new password one more time"
  CharsetSourceEnc IBM-1047
  CharsetDefault ISO8859-1
</Location>
```

The Location directive tells IBM HTTP Server that any URL that is received that ends with `sdsfViewer.rx` is to feature the directives that are specified within that Location block of directives that are applied.

You can change the value for the `AuthName` directive; for example, it might be `Company ABC z/OS Viewer`. You must restart the V8.5.5 server to pick up this change. When accessing this Rexx, you are prompted for a user ID and password, as shown in Figure 7-1.

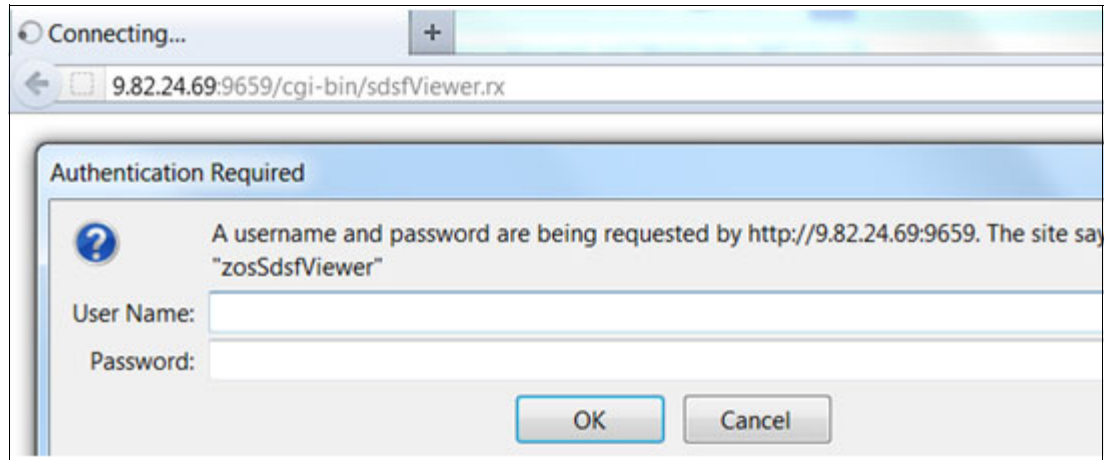


Figure 7-1 RACF user ID and password prompt

7.2.3 Allowing authenticated user that belongs to a group access

The next level of access control allows only users who supply a user ID and password that are validated by using the z/OS SAF interface and who are a member of a designated group to access spool by using the Rexx program. This approach restricts access to those authenticated users who are members of a defined RACF group.

This level of controlled access is HTTP Server enforced. We are still in the client-as-server mode, where the SDSF/JES spool access is done under the user ID of the HTTP Server. To achieve this access, the modifications to the V8.5.5 `httpd.conf` file that are described next are required.

Add the `Require saf-group` directive to the Location block of directives, which now looks similar to what is shown in Example 7-3.

Example 7-3 Add saf-group on httpd.conf file example

```
<Location /waslogs/>
  CharsetSourceEnc ISO8859-1
  AddEncoding x-gzip gz tgz
  Header set Content-Disposition "attachment;"
  AuthName zosSdsfViewer
  AuthType Basic
  AuthBasicProvider saf
  Require saf-group E1CELL
  AuthSAFExpiration "EXPIRED! oldpw/newpw/newpw"
  AuthSAFReEnter "Enter new password one more time"
</Location>
```

Note: You cannot use `Require valid-user` when `Require saf-group` is used.

Restart the V8.5.5 server. When a user accesses the URL to run the Rexx program, they are prompted for their user ID and password. IHS first validates the user ID and password with RACF. Then, V8.5.5 validates that the user ID is connected to the group that is specified on the directive.

Note: Multiple group names can be specified on the directive.

7.2.4 Allowing authenticated user access with client credentials

The next level of access control is to configure V8.5.5 so that it uses the authenticated and authorized client user ID to access the JES Spool rather than the user ID of the STC the V8.5.5 is running under. This setup provides more granular control over which STC output a user that uses this Rexx can view.

Although this configuration results in the V8.5.5 accessing the JES Spool by using the authenticated user ID, the STC output they can view depends on how security was set up in SDSF and SAF. To achieve this access, modifications to the V8.5.5 httpd.conf file are required. Add the SAFRunAS directive to the Location block of directives, which resembles Example 7-4.

Example 7-4 Adding SAFRunAS on the httpd.conf file example

```
<Location /waslogs/>
  CharsetSourceEnc ISO8859-1
  AddEncoding x-gzip gz tgz
  Header set Content-Disposition "attachment;"
  AuthName zosSdsfViewer
  AuthType Basic
  AuthBasicProvider saf
  Require saf-group E1CFG
  SAFRunAS %%CLIENT%%
  AuthSAFExpiration "EXPIRED! oldpw/newpw/newpw"
  AuthSAFReEnter "Enter new password one more time"
</Location>
```

When V8.5.5 receives a request, it validates the RACF user ID and password. When the Rexx program runs and attempts to access SDSF, the user ID that this access occurs under is the validated RACF user ID.

Note: In Example 7-4, we used `Require saf-group`, but you can also use `Require valid-user`, depending on how you want to control access.

7.2.5 Required SAF definitions

To use SAFRunAS, you must update RACF (or whatever security product is used) to use SAFRunAS. The following RACF commands are required:

```
RDEFINE FACILITY BPX.SERVER UACC(NONE) (if not defined already)
PERMIT BPX.SERVER CLASS(FACILITY) ID(ihs_user_id) ACC(read)
SETROPTS RACLIST(FACILITY) REFRESH
```

If you do not set up RACF rules, you see a message in the syslog when you try to run the Rexx program, as shown in Example 7-5.

Example 7-5 Syslog message example

```
ICH408I USER(WEBSRV1 ) GROUP(SYS1      ) NAME(WEBSRV1          )
      BPX.SERVER CL(FACILITY)
      INSUFFICIENT ACCESS AUTHORITY
      ACCESS INTENT(READ  ) ACCESS ALLOWED(NONE  )
```

Note: The user IDs of the users must have a defined OMVS segment.

7.2.6 Complex authorization logic

IBM HTTP Server powered by Apache V9.0 is built on Apache V2.4, which provides enhanced capability for controlling authorization. New directives allow you to combine the authorization directives to set up complex authorization logic. For more information about these new directives, see this website:

http://httpd.apache.org/docs/2.4/mod/mod_authz_core.html#logic

Figure 7-2 shows an example of how a complex authorization is set up.

```
<Directory /www/mydocs>
  <RequireAll>
    <RequireAny>
      Require user superadmin
    <RequireAll>
      Require group admins
      Require ldap-group cn=Administrators,o=Airius
    <RequireAny>
      Require group sales
      Require ldap-attribute dept="sales"
    </RequireAny>
  </RequireAll>
</RequireAny>
<RequireNone>
  Require group temps
  Require ldap-group cn=Temporary Employees,o=Airius
</RequireNone>
</RequireAll>
</Directory>
```

Figure 7-2 Example of complex authorization

7.3 SSL and Session ID

SSL provides for secure transmission of data across Internet Protocol networks. However, this secure transmission is costly because of the extra CPU that is used to establish SSL connections and the associated encryption and decryption.

z/OS provides a mechanism (which is referred to as the *SID cache*) that can help avoid the extra processing of having to reestablish an SSL connection. For more information, see this website:

<https://ibm.biz/Bdr59Q>

7.4 Configuring SSL support

Setting up your IBM HTTP Server powered by Apache to use SSL requires the use of various SSL-related directives and setting up certificates. This section describes the commands that are used to enable IBM HTTP Server powered by Apache to support the use of SSL.

7.4.1 RACF or keystore files

Store your digital certificates in RACF. This approach is more secure than the alternative of the use of keystore files in IBM HTTP Server powered by Apache. Leaving certificates in keystore files in the z/OS UNIX environment is not considered as robust as the use of RACF from a security perspective.

7.4.2 Creating required certificates

Example 7-6 shows the RACF commands that were used to create the required certificates and key rings.

Example 7-6 Setting up certificates and key ring

* Create a certificate to sign the server certificate

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('IHS CertAuth') OU('IHS RedPaper')) ,  
WITHLABEL('IHS.Redpaper') TRUST SIZE(1024) NOTAFTER(DATE(2021/12/31))
```

* Create a server side certificate

```
RACDCERT ID (IHSAESTC) GENCERT SUBJECTSDN(CN('wtsc55.itso.ibm.com') O('IBM')  
OU('IHS')), WITHLABEL('IHS'), SIGNWITH(CERTAUTH LABEL('IHS.Redpaper')) ,  
SIZE(1024), NOTAFTER(DATE(2021/12/31))
```

* Create a key ring for the userid the server runs under

```
RACDCERT ADDRING(IHSKeyring.ITSO) ID(IHSAESTC)
```

* Connect server certificate to user keyring

```
RACDCERT ID(IHSAESTC) CONNECT (LABEL('IHS') RING(IHSKeyring.ITSO) DEFAULT)
```

* Connect singer certificate to user keyring '

```
RACDCERT ID(IHSAESTC) CONNECT (RING(IHSKeyring.ITSO) LABEL('IHS.Redpaper')  
CERTAUTH)
```

```
SETOPTS RACLIST(DIGTCERT) REFRESH
```

We then issued the following command to give the User ID that the IHS server runs under access to the certificates that were created in Example 7-6:

```
permit IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(IHSAESTC) ACCESS(READ)  
SETOPTS RACLIST(facility) REFRESH
```

If this access is not permitted, a RACF violation message is shown that is similar to the following example when you start the server:

```
ICH408I USER(IHSAESTC) GROUP(IHSRB13 ) NAME(IHS SERVER 1 EDWARD )  
IRR.DIGTCERT.LISTRING CL(FACILITY)  
INSUFFICIENT ACCESS AUTHORITY  
ACCESS INTENT(READ ) ACCESS ALLOWED(NONE )
```

7.4.3 Updating httpd.conf

The default SSL directives in the httpd.conf file when a server is first created are shown in Example 7-7.

Example 7-7 Default SSL settings

```
#LoadModule ibm_ssl_module modules/mod_ibm_ssl.so  
#Listen 443  
#<VirtualHost *:443>  
#SSLEnable  
#</VirtualHost>  
#KeyFile /ihsconfig/ihs/ihsae002/ihsserverkey.kdb
```

The changes that were made to enable SSL support are shown in Example 7-8.

Example 7-8 Enabling SSL support

```
LoadModule ibm_ssl_module modules/mod_ibm_ssl.so  
Listen 8236  
<VirtualHost *:8236>  
SSLEnable  
</VirtualHost>  
KeyFile /saf IHSKeyring.ITS0
```

The server must be stopped and started to pick up this change. The SSL-related messages that appeared in the server error_log file are shown in Example 7-9.

Example 7-9 SSL-related messages from error_log

```
[Thu Jul 11 04:36:59 2013] [info] SSL0320I: Using SSLV3 Cipher:  
TLS_RSA_WITH_AES_128_CBC_SHA(2F)  
[Thu Jul 11 04:36:59 2013] [info] SSL0320I: Using SSLV3 Cipher:  
TLS_RSA_WITH_AES_256_CBC_SHA(35b)  
[Thu Jul 11 04:36:59 2013] [info] SSL0320I: Using SSLV3 Cipher:  
SSL_RSA_WITH_RC4_128_SHA(35)  
[Thu Jul 11 04:36:59 2013] [info] SSL0320I: Using SSLV3 Cipher:  
SSL_RSA_WITH_RC4_128_MD5(34)  
[Thu Jul 11 04:36:59 2013] [info] SSL0320I: Using SSLV3 Cipher:  
SSL_RSA_WITH_3DES_EDE_CBC_SHA(3A)  
[Thu Jul 11 04:36:59 2013] [info] SSL0320I: Using TLSv10 Cipher:  
TLS_RSA_WITH_AES_128_CBC_SHA(2F)  
[Thu Jul 11 04:36:59 2013] [info] SSL0320I: Using TLSv10 Cipher:  
TLS_RSA_WITH_AES_256_CBC_SHA(35b)  
[Thu Jul 11 04:36:59 2013] [info] SSL0320I: Using TLSv10 Cipher:  
SSL_RSA_WITH_RC4_128_SHA(35)  
[Thu Jul 11 04:36:59 2013] [info] SSL0320I: Using TLSv10 Cipher:  
SSL_RSA_WITH_RC4_128_MD5(34)
```

```
[Thu Jul 11 04:36:59 2013] [info] SSL0320I: Using TLSv10 Cipher:
SSL_RSA_WITH_3DES_EDE_CBC_SHA(3A)
[Thu Jul 11 04:36:59 2013] [debug] mod_ibm_ssl.c(4014): Empty cipher list
specified for SSLV2, disabling
[Thu Jul 11 04:36:59 2013] [debug] mod_ibm_ssl.c(4025): Setting ciphers for SSLV3
to "002F003500050004000A"
[Thu Jul 11 04:36:59 2013] [debug] mod_ibm_ssl.c(4025): Setting ciphers for TLSv10
to "002F003500050004000A"
[Thu Jul 11 04:36:59 2013] [debug] mod_ibm_ssl.c(4041): TLSv11 disabled, not
setting ciphers
[Thu Jul 11 04:36:59 2013] [debug] mod_ibm_ssl.c(4041): TLSv12 disabled, not
setting ciphers
```

The SSL-related messages that were output in the `error_log` file when we start an IHS V9.0 server on z/OS 2.2 are shown in Example 7-10.

Example 7-10

```
System SSL: SHA-1 crypto assist is available
System SSL: SHA-224 crypto assist is available
System SSL: SHA-256 crypto assist is available
System SSL: SHA-384 crypto assist is available
System SSL: SHA-512 crypto assist is available
System SSL: DES crypto assist is available
System SSL: DES3 crypto assist is available
System SSL: AES 128-bit crypto assist is available
System SSL: AES 256-bit crypto assist is available
System SSL: AES-GCM crypto assist is available
System SSL: Cryptographic accelerator is not available
System SSL: Cryptographic coprocessor is not available
System SSL: Public key hardware support is not available
System SSL: ECC secure key support is not available.
System SSL: ICSF Secure key PKCS11 support is not available
```

7.4.4 Testing SSL

We then entered the following URL in a browser:

`https://wtsc55.itso.ibm.com:8236`

The browser displayed a warning message in which it was advised that the server certificate was not signed by a recognized certificate authority. This issue was expected, so it is accepted.

We see that the SSL connection was established in the browser and the home page was displayed.

The HTTP Server uses the z/OS `gskkyman` tool for key management to create a CMS key database file, public and private key pairs, and self-signed certificates. You also can create a SAF key ring in place of a CMS key database file.

For more information about `gskkyman`, see this website:

<https://ibm.biz/Bdr5CR>

For more information about creating SAF key rings, see this website:

<https://ibm.biz/Bdr5CX>

7.4.5 Advanced SSL options

You can enable advanced security options, such as client authentication, setting and viewing cipher specifications, defining SSL for multiple-IP virtual hosts, and setting up a reverse proxy configuration with SSL. For more information, see this website:

<https://ibm.biz/Bdr5Cr>

7.4.6 Basic SNI Support

Basic SNI support provides a way to allow a single server to present different server side site certificates for the same TCP/IP address.

What problem does basic SNI support solve?

An IBM HTTP Server powered by Apache is shown in Figure 7-3. It is listening on port 443, which was configured to be the port to handle SSL connections to the server.

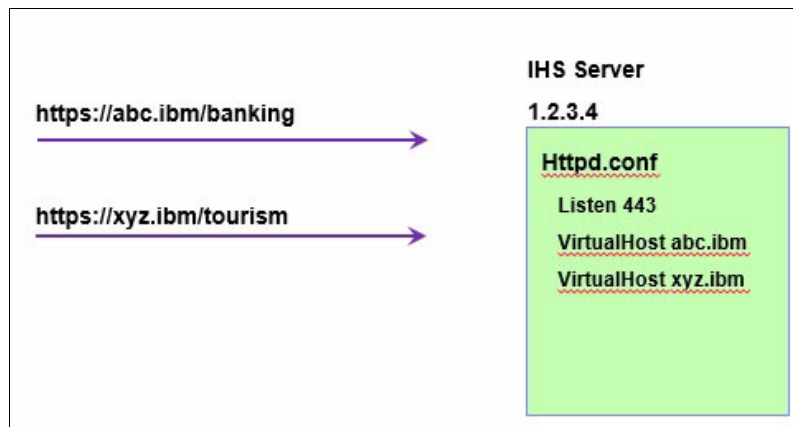


Figure 7-3 Basic SNI problem

The `httpd.conf` file was set up to support two virtual hosts (`abc.ibm` and `xyz.ibm`). The TCP/IP address of the server is `1.2.3.4`. The DNS names `abc.ibm` and `xyz.ibm` resolve to this same address.

The `httpd.conf` file also is configured to send a server side certificate when an SSL connection is established.

However, before IBM HTTP Server powered by Apache V9, only one SSL certificate was presented by the server to all users. This certificate's DNS name was set to `n=abc.ibm,o=ibm`.

When users access the `abc.ibm` site, they are presented with a certificate with a DNS name that matches the DNS name of the site they are accessing. For these users, all is OK.

However, the users who access the xyz.ibm site also are presented with the same certificate. In these days of heightened awareness of hacking through the Internet, a user who is accessing xyz.ibm and then sees a certificate with a different DNS name of abc.ibm can incorrectly think something is amiss. This issue can lead to confusion and dissatisfaction with the site.

Resolving the problem

IBM HTTP Server powered by Apache V9 provides a way to allow the server to present different server side certificates, depending on what DNS name the server receives.

This presentation is done by the SNI support that is added in the underlying Apache 2.4 code.

SNI allows you to configure the IBM HTTP Server powered by Apache V9 so that it can present different server side certificates for different DNS names that are hosted on the same TCP/IP address.

For more information about the changes to the directive to support SNI, see this website:

http://publib.boulder.ibm.com/httperv/manual24/mod/mod_ibm_ssl.html#SNI

Implementing SNI example

To test SNI, we issued the RACF commands that are shown in Example 7-11 to create a certificate and attach it to the keyring that we set up in 7.4.2, “Creating required certificates” on page 123.

Example 7-11 Defining new server side certificate

```
RACDCERT ID (IHSAESTC) GENCERT SUBJECTSDN(CN('pok55.itso.ibm.com') O('IBM')
OU('IHS')), WITHLABEL('IHSPOK55'), SIGNWITH(CERTAUTH LABEL('IHS.Redpaper')) ,
SIZE(1024), NOTAFTER(DATE(2021/12/31))

RACDCERT ID(IHSAESTC) CONNECT (LABEL('IHSPOK55') RING(IHSKeyring.ITS0) )

SETROPTS RACLIST(DIGTCERT) REFRESH
```

We then changed the httpd.conf file to use SNI by adding the lines that are shown in Example 7-12.

Example 7-12 Directives to support SNI

```
<VirtualHost *:8236>
    SSLEnable SNI
    ServerName itso.ibm.com
    SSLSNIMap wtsc55.itso.ibm.com IHS
    SSLSNIMap pok55.itso.ibm.com IHSPOK55
</VirtualHost>

KeyFile /saf IHSKeyring.ITS0
```

We restarted the server and then entered the following URLs:

```
https://wtsc55.itso.ibm.com:8236/
https://pok55.itso.ibm.com:8236/
```

In the browser, we viewed the received server side certificate for each site and saw that we received the correct certificate for each site.

An alternative method to achieve the same result is shown in Example 7-13.

Example 7-13 Alternative method to support SNI

```
<virtualhost *:8236>
  ServerName itso.ibm.com
  SSLEnable SNI
</virtualhost>
<virtualhost *:8236>
  ServerName pok55.itso.ibm.com
  SSLEnable
  SSLServerCert IHSP0K55
</virtualhost>
```

7.5 Controlling access by using mod_rewrite

Apache provides a module called `mod_rewrite`. The directives in this module provide a powerful capability you can use in V8.5.5 to control which requests are processed. An advantage of the use of `mod_rewrite` directives is that they help to ensure that only the URLs you expect are processed. They also help to prevent unauthorized access to files within your V8.5.5 that you do not want to be accessed.

For more information, see the following resources:

- ▶ <http://publib.boulder.ibm.com/httserv/manual70/misc/rewriteguide.html>
- ▶ <https://httpd.apache.org/docs/2.2/rewrite/access.html>

To help get you started on understanding the use of `mod_rewrite`, this section describes how you can ensure that only a specific program in the `cgi-bin` directory can be started. The URL that is used to start a Rexx that is named `sdsfViewer` with the default setup looks like the following example:

```
http://wsc1.washington.ibm.com:9659/cgi-bin/sdsfViewer.rx
```

The default `httpd.conf` file allows this Rexx and any other program in the `cgi-bin` directory to be run. Our aim is to show how (with a few changes) we can ensure that only the `sdsfViewer.rx` CGI program can be accessed, and only the following URL works:

```
http://wsc1.washington.ibm.com:9659/tools/sdsfViewer.rx
```

First, we must make the `mod_rewrite` module available to the V8.5.5 server. The following group of directives is found in `httpd.conf`:

```
#LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
#LoadModule rewrite_module modules/mod_rewrite.so
#LoadModule deflate_module modules/mod_deflate.so
```

Remove the `#` on the `LoadModule` directive for the `rewrite_module` to uncomment it so that the code now looks like the following example:

```
#LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so
#LoadModule deflate_module modules/mod_deflate.so
```

We then want to use the word “tools” in the URL rather than “cgi-bin.” The following line must be included in the `httpd.conf` file:

```
ScriptAlias /cgi-bin/ "/shared/ih_sdsf/cgi-bin/"
```

We added the following directive after the previous directive:

```
ScriptAlias /tools/ "/shared/ih_sdsf/cgi-bin/"
```

This directive defines the alias name `tools` that maps to the `cgi-bin` directory.

We then use the `ReWriteCond` and `ReWriteRule` directives to ensure that only the `sdsfViewer.rx` CGI program can be accessed. At the bottom of the `httpd.conf` file, add the following code:

```
<VirtualHost *:9659>
RewriteEngine On
# Only Allow requests associated with the sdsfViewer Rexx
# If not expected URI, then redirect to URL to start the Rexx
RewriteCond %{REQUEST_URI}
!(/tools/sdsfViewer.rx$|/icons/back.gif$|icons/compressed.gif$|images/zec12.jpg$)
RewriteRule .* http://wscl.washington.ibm.com:9659/tools/sdsfViewer.rx [R,L]
</VirtualHost>
```

This `ReWriteCond` directive is an IF test to see whether the received URL matches any of the specified values. If the URLs match the values, the request is processed. If the URLs do not match, a redirect request is sent to the browser, which results in the browser sending a request to start the `sdsfViewer Rexx`.

You must modify the `ReWriteRule` to reflect the DNS name of your site. The `ReWriteCond` directive also must be modified if you use a different value for the value of the variable `sdsfRexxPath` in the `sdsfViewer.rx` program.

The net result of adding this code is that IBM HTTP Server can be used only to run the `sdsfViewer Rexx`.

This description is an example of the usefulness of `mod_rewrite` in controlling the requests that the V8.5.5 processes. Many other methods can be used in which you `mod_rewrite` to suit your requirements.

7.6 Caching and security considerations

This section describes caching and security considerations.

7.6.1 Authorization and access control

The use of `mod_cache` is much like having a built-in reverse-proxy. Requests are served by the caching module unless it determines that the back-end must be queried. Caching local resources drastically changes the security model of V8.5.5.

Because traversing a file system hierarchy to examine potential `.htaccess` files is an expensive operation that partially defeats the point of caching (to speed up requests), `mod_cache` makes no decision about whether a cached entity is authorized for serving. That is, if `mod_cache` cached some content, it is served from the cache if that content did not expire.

For example, if your configuration permits access to a resource by IP address, you must ensure that this content is not cached. You prevent this cache by using the `CacheDisable` directive or `mod_expires`. Left unchecked, `mod_cache` (much like a reverse proxy) caches the content when served and then serves it to any client on any IP address.

7.6.2 Local vulnerabilities

As requests to users can be served from the cache, the cache can become a target for those authorized users who want to deface or interfere with content. It is important to remember that the cache must always be writable by the user on which V8.5.5 is running. This requirement is in stark contrast to the popular recommendation that all content that is unwritable by the HTTP Server user is maintained.

If the server user is compromised (for example, through a flaw in a CGI process), the cache might be targeted. When `mod_disk_cache` is used, it is relatively easy to insert or modify a cached entity.

This issue presents an elevated risk in comparison to the other types of attacks that are possible to make as the server user. If `mod_disk_cache` is used, you must be aware of this risk. Ensure that you upgrade your server when security upgrades are announced and run CGI processes as a non-server user by using `suEXEC`, if possible.

7.6.3 Cache poisoning

When V8.5.5 is running as a caching proxy server, the potential for so-called cache poisoning exists. This broad term is often used for attacks in which an attacker causes the proxy server to retrieve incorrect (and often unwanted) content from the back-end.

For example, if the DNS servers that are used by your system are vulnerable to DNS cache poisoning, an attacker might be able to control where V8.5.5 connects to when requesting content from the origin server. Another example is the so-called HTTP request-smuggling attack.

This Redbooks publication does not present an in-depth description of HTTP request smuggling. However, it is important to be aware that a series of requests can be made, a vulnerability can be used on an origin web server such that the attacker can entirely control the content that is retrieved by the proxy.



System Management Facilities support in IHS V8.5.5

This chapter describes the System Management Facilities (SMF) support that is now available in V8.5.5 of IHS powered by Apache and compares it to V5.3 of IHS powered by Domino.

This chapter includes the following topics:

- ▶ 8.1, “SMF overview” on page 132
- ▶ 8.2, “DGW and SMF” on page 132
- ▶ 8.3, “V8.5.5 and SMF” on page 132
- ▶ 8.4, “Summary” on page 136

8.1 SMF overview

SMF is a component of z/OS that is used to record various detailed information about resources that are used by processes that run on z/OS. For more information about SMF, see this website:

<https://ibm.biz/Bdr57N>

DGW featured support to produce SMF records for many years. V8.5.5 of the HTTP Server powered by Apache introduces support to allow it to also produce SMF records.

If you have an older version of IBM HTTP Server powered by Apache and want to obtain SMF records, the following IBM Techdoc provides a sample module that includes some of this functionality:

<https://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101225>

8.2 DGW and SMF

DGW writes SMF records of type 103. You can configure DGW to produce the following subtypes of SMF type 103 data:

- ▶ Subtype 1: Configuration information
- ▶ Subtype 2: Performance information

For more information about the information that is recorded in these subtypes, see this website:

<https://ibm.biz/Bdr5WD>

For more information about how to enable DGW to produce SMF records, see this website:

<https://ibm.biz/Bdr5WF>

8.3 V8.5.5 and SMF

V8.5.5 also writes SMF records of type 103. You can configure V8.5.5 to produce the following subtypes:

- ▶ Subtype 13: Thread statistics
- ▶ Subtype 14: Information about each request

8.3.1 Comparing DGW and V8.5.5 SMF records

The key difference between the information that is recorded by V8.5.5 and DGW is that V8.5.5 provides the capability to record an SMF record for each processed request whereas DGW does not. The subtype 14 SMF record shows how much CPU was used to process an individual request.

8.3.2 Content

The data that is collected in the subtype 13 SMF records is listed in Table 8-1. This data is periodic thread statistics that are produced by the `mod_mpmstats` module.

Table 8-1 Type 103 subtype 13 record format

Field	Length
parent process ID	4
ready threads	4
busy threads	4
reading threads	4
writing threads	4
logging threads	4
dns threads	4
closing threads	4
keepalive threads	4
bytes served	8
requests served	8
servername len	4
servername	\$servername_len

The `mod_mpmstats` module records periodic information about IBM HTTP Server processing threads in the error log. Before V7R0, it was available only as part of the `ihdiag mustgather` tool.

In V8.5.0.0 and later, the displayed number of threads in keepalive state is always zero because of the Event MPM. Threads that are busy in non-module code cannot be counted with `TrackModules`. For more information, see this website:

http://wilson.boulder.ibm.com/httperv/manual70/mod/mod_mpmstats.html

The data that collected in the subtype 14 SMF records is listed in Table 8-2. This information is HTTP access log data that is produced by the `mod_smf` module.

Table 8-2 Type 103 subtype 14 record format

Field	Offset
process ID	0-3
method length (from start of data buffer)	4-7
domain length (from end of method)	8-11
uri length (from end of domain)	12-15
remote_ip length (from end of remote_ip)	16-19
cpu time	20-27
lapsed time (string)	28-35
variable length buffer	26+

For more information, see this website:

http://wilson.boulder.ibm.com/httperv/manual70/mod/mod_smf.html

8.3.3 SMF browser

IBM supplies an SMF browser that uses a JAR file that is named `bbomsmfv`, which can be used to display the SMF record content in a formatted way. It is available at this website:

<https://www.software.ibm.com/webapp/iwm/web/preLogin.do?source=zosos390>

We used this browser to display the SMF records that we collected to show the content of the subtype 13 and 14 SMF records.

8.3.4 Enabling for subtype 13

Enabling V8.5.5 to produce subtype 13 SMF records requires that the `mpmstats` module is loaded and that the `SMFReportInterval` directive is specified. The supplied `httpd.conf` includes the `mpmstats` module that is enabled by default as shown in the following example:

```
# mod_mpmstats logs statistics about server activity to the main
# error log. No records are written while the server is idle.
LoadModule mpmstats_module modules/debug/mod_mpmstats.so
<IfModule mod_mpmstats.c>
# Write a record every 10 minutes (if server isn't idle).
# Recommendation: Lower this interval to 60 seconds, which will
# result in the error log growing faster but with more accurate
# information about server load.
ReportInterval 600
# Include details of active module in the statistics.
TrackModules On
</IfModule>
```

The only parameter for the `SMFReportInterval` directive is a number that indicates the interval between the writing of the subtype 13 SMF records. You can add the directive after the `ReportInterval`, as shown in the following example:

```
ReportInterval 600
SMFReportInterval 600
```

Sample

After the V8.5.5 was running for some time, we ran the following JCL to extract the subtype 13 SMF 103 records:

```
//IHS103    EXEC   PGM=IFASMFDP
//DUMPIN    DD     DSN=SYS1.SC55.MAN2,DISP=SHR
//DUMPOUT   DD     DSN=EDMCAR.SMFDATA.IHS103,DISP=(NEW,CATLG),
//           SPACE=(CYL,(10,10)),DCB=(RECFM=VB,LRECL=32760)
//SYSPRINT  DD     SYSOUT=*
//SYSIN     DD     *
              INDD(DUMPIN,OPTIONS(DUMP)),
              OUTDD(DUMPOUT,TYPE(103(13)))
```

We issued the following command to format the collected SMF records:

```
java -cp bbomsmfv.jar:batchsmf.jar com.ibm.ws390.sm.smfview.SMF
'INFILE(EDMCAR.SMFDATA.IHS103)' 'PLUGIN(DEFAULT,STDOUT)' > ihsSmf.txt
```


The formatted output for a subtype 13 SMF record is shown in the following example:

```
Record#: 2141;
  Type: 103; Size: 100; Date: Sat Jun 22 07:19:15 EDT 2013;
  SystemID: SC55; SubsystemID: null; Flag: 94;
  Subtype: 13 (Unknown SMF Record type/subtype combination);
ServerName: wtsc55.itso.ibm.com; PID: 33688136;
Thread Details: [ ready: 2; busy: 6; reading: 0; writing: 6; logging: 0; dns: 0;
closing: 0; keepalive: 0; ]
Cumulative: [ kbytes: 250; requests: 1281; ]
```

In the output, we noticed that the description for the subtype 13 is shown as unknown. We reported this issue to the author of the bbomsmfv JAR file and expect to be updated in the future.

8.3.5 Enabling for subtype 14

To enable V8.5.5 to write subtype 14 SMF records, add a directive to inform V8.5.5 to load the SMF module. Load the mod_smf module and choose a context to enable SMF logging with the following SMFRecord directive:

```
LoadModule smf_module modules/mod_smf.so
...
<Location /IBMTools/>
SMFRecord ON
</Location>
```

Sample

After sending some requests to the V8.5.5, we ran this JCL to extract the subtype 14 SMF 103 records:

```
//IHS103    EXEC   PGM=IFASMFDP
//DUMPIN    DD     DSN=SYS1.SC55.MAN2,DISP=SHR
//DUMPOUT   DD     DSN=EDMCAR.SMFDATA.IHS103,DISP=(NEW,CATLG),
//          SPACE=(CYL,(10,10)),DCB=(RECFM=VB,LRECL=32760)
//SYSPRINT  DD     SYSOUT=*
//SYSIN     DD     *
             INDD(DUMPIN,OPTIONS(DUMP)),
             OUTDD(DUMPOUT,TYPE(103(14)))
```

We issued the following command to format the collected SMF records:

```
java -cp bbomsmfv.jar:batchsmf.jar com.ibm.ws390.sm.smfview.SMF
'INFILE(EDMCAR.SMFDATA.IHS103)' 'PLUGIN(DEFAULT,STDOUT)' > ihsSmf.txt
```

The formatted output for a subtype 14 SMF record is shown in the following example:

```
Record#: 101;
  Type: 103; Size: 119; Date: Sat Jun 22 08:22:07 EDT 2013;
  SystemID: SC55; SubsystemID: STC; Flag: 94;
  Subtype: 14 (Unknown SMF Record type/subtype combination);
pid=84019836 method=GET host=w3.sc55.itso.ibm.com:8235 uri=/IBMTools/Sleeper rip =
9.190.237.212 elapsed= 2010 cpu=0.00032
```

SMF debug directive

A directive to get V8.5.5 output debug information about the processing of subtype 14 SMF records can be added to the V8.5.5 httpd.conf file. Add the following line near the bottom of the httpd.conf file:

```
SMFLogDebug ON
```

After directive is added and V8.5.5 is restarted, we sent one request to the server and observed the following messages in the error_log file:

```
[Sun Jun 23 20:08:32 2013] [debug] mod_smf.c(321): SMF record content in hex and char:
[Sun Jun 23 20:08:32 2013] [debug] mod_smf.c(360): 007C00005E67006EA5010113173FE2E8
.@..;..>v.....SY
[Sun Jun 23 20:08:32 2013] [debug] mod_smf.c(360): E2C5E2E3C340000E05020BA000000003
SESTC.....
[Sun Jun 23 20:08:32 2013] [debug] mod_smf.c(360): 0000001A0000001700000000CF04BF0F0
.....0.00
[Sun Jun 23 20:08:32 2013] [debug] mod_smf.c(360): F0F5F30000000000000000FC7C5E3A6
053.....GETw
[Sun Jun 23 20:08:32 2013] [debug] mod_smf.c(360): A3A283F5F596854B89A3A2964B898294
tsc55oe.itso.ibm
[Sun Jun 23 20:08:32 2013] [debug] mod_smf.c(360): 4B8396947AF8F2F3F561C9A3A296E396
.com:8235/ItsoTo
[Sun Jun 23 20:08:32 2013] [debug] mod_smf.c(360): 9693A261C5C289A9C889A3C396A495A3
ols/EBizHitCount
[Sun Jun 23 20:08:32 2013] [debug] mod_smf.c(360): F94BF1F9F04BF1F6F54BF8F2
9.190.165.82
[Sun Jun 23 20:08:32 2013] [info] in SMF: write RC.: 0
```

8.4 Summary

Many clients that use z/OS often have a process in place that uses SMF records to analyze the consumption of resources and performance of their various systems and applications. The introduction of support in V8.5.5 to produce SMF records provides a way for these clients to add V8.5.5 to this process. For those clients who do not have such a process, the SMF support is still valuable in analyzing how V8.5.5 is performing daily.



Plug-in for WebSphere Application Server

This chapter compares the setup of the WebSphere Application Server plug-in for the two IBM HTTP Servers that run on z/OS.

WebSphere Application Server for z/OS provides a plug-in to use with IBM HTTP Server on all platforms. Because two IBM HTTP Servers are available on z/OS, a version that is suitable for use with each IBM HTTP Server is provided.

The plug-in performs the same purpose in both IBM HTTP Servers, which is to proxy requests they receive to the back-end WebSphere Application Servers.

It is assumed that the WebSphere Application Server plug-in code was installed on the z/OS LPAR.

This chapter includes the following topics:

- ▶ 9.1, “Plug-in overview” on page 138
- ▶ 9.2, “Intelligent Management for Web Servers feature” on page 138
- ▶ 9.3, “Configuring WebSphere Application Server plug-in into IBM HTTP Servers” on page 140

9.1 Plug-in overview

The WebSphere Application Server plug-in works much the same in both IBM HTTP Servers on z/OS.

The web server plug-in uses an XML configuration file to determine whether a request is for the web server or the application server. This XML configuration file can be generated by the WebSphere Application Server Administration console or by using **wsadmin** commands. For more information, see the WebSphere Application Server Knowledge Center at this website:

<https://ibm.biz/BdrNGK>

The basic operation of the plug-in for IBM HTTP Server is that when a request reaches the web server, the URL is compared to the URLs that are managed by the plug-in. If a match is found, the plug-in configuration file contains the information that is needed to forward that request to the web container by using the web container inbound transport chain, as shown in Figure 9-1.

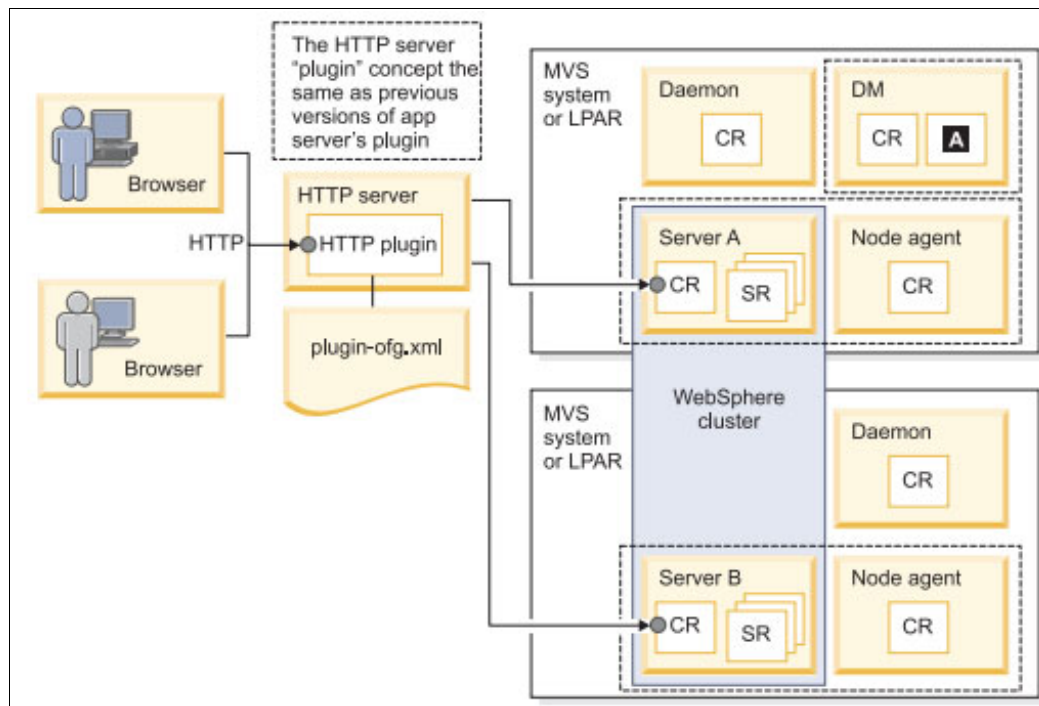


Figure 9-1 Web server plug-in routing

When you change the WebSphere Application Server configuration, how requests are routed from a web server to the application server is affected. You must regenerate and propagate the plug-in configuration file to the web server. You can propagate the file manually or configure the propagation to be done automatically.

9.2 Intelligent Management for Web Servers feature

The Intelligent Management for Web Servers feature for WebSphere Application Server is new in V8.5.5. This feature simplifies intelligent management capabilities through integration of the On Demand Router (ODR) capability into the WebSphere Web Server plug-in to reduce the need for the separate Java based proxy server in many scenarios.

Before this release, the typical method that was used to implement the ODR in your environment is shown in Figure 9-2. This method was used because the ODR is a Java based component and there were security concerns regarding the use of an ODR in a DMZ.

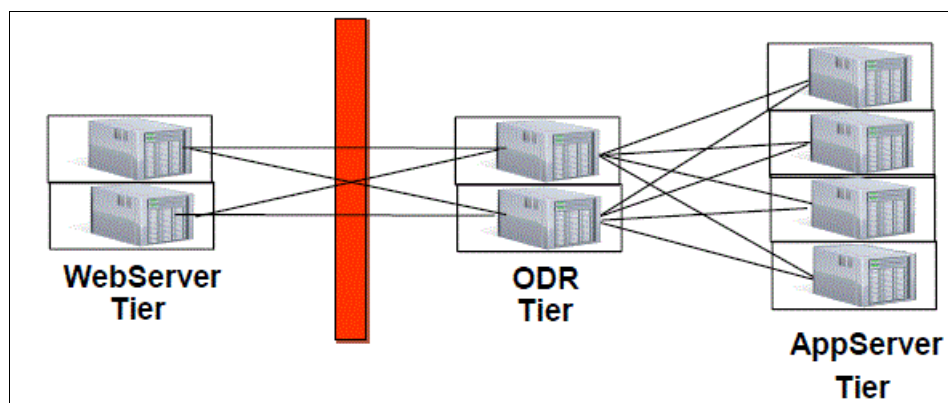


Figure 9-2 Previous way to use ODR

In WebSphere Application Server V8.5.5, the ODR is now available as a plug-in to run in an IBM HTTP Server and is written in C code. This approach allows the ODR plug-in to be used in IBM HTTP Servers, as shown in Figure 9-3.

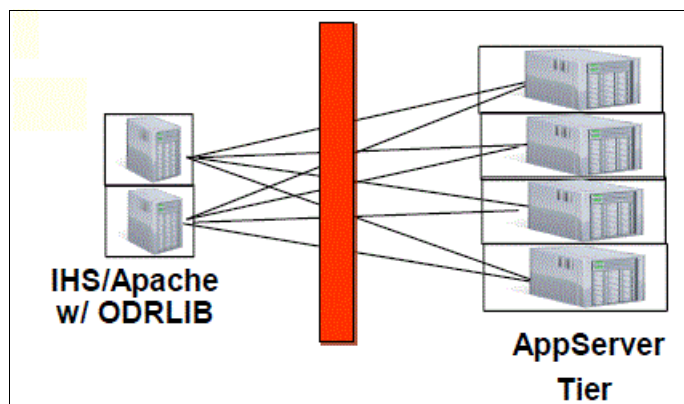


Figure 9-3 New approach for using ODR

Clients often run IBM HTTP Servers on distributed operating systems in the DMZ. Because these devices are on the front line in attempting to defend your site from hacking attempts, consider the use of IBM HTTP Servers on z/OS in the DMZ to provide a more secure operating system to help with this task.

The ODR is the WebSphere routing function that classifies incoming requests and then dispatches the requests across the application server environment. This integration provides significant value in terms of consolidation and simplification of the topology, which resulting in fewer resources to manage and lower total cost of ownership.

Avoiding a problem: On WebSphere Application Server for z/OS, web servers with Intelligent Management enabled do not start. For more information about how to resolve this situation, see this Technote:

<http://www.ibm.com/support/docview.wss?uid=swg21636467>

The older ODR approach is stabilized in WebSphere Application Server V8.5.5, as described at this website:

<https://ibm.biz/BdrNGJ>

9.3 Configuring WebSphere Application Server plug-in into IBM HTTP Servers

The methods that are available to configure the WebSphere Application Server plug-in into IBM HTTP Servers and the related directives are different. The WebSphere Application Server plug-in code supplies a version of the plug-in module to use for the two different IBM HTTP Servers that run on z/OS.

9.3.1 IBM HTTP Server powered by Domino

For more information about how to configure the WebSphere Application Server plug-in into an IBM HTTP Server powered by Domino, see this website:

<https://ibm.biz/BdrNGu>

The process to set up IBM HTTP Server powered by Domino is manual. You must add at least three directives, as shown in Example 9-1.

Example 9-1 Examples of directives that are used to use WebSphere Application Server plug-in

```
ServerInit
/usr/lpp/zWebSphereEM1_Plugins/V8R5/bin/ih390WASPlugin_http.so:init_exit
/ih3config/dws/ih3de001/plugin/plugin-cfg-ih3de001.xml
Service /IBMTTools/*
/usr/lpp/zWebSphereEM1_Plugins/V8R5/bin/ih390WASPlugin_http.so:service_exit
ServerTerm
/usr/lpp/zWebSphereEM1_Plugins/V8R5/bin/ih390WASPlugin_http.so:term_exit
```

Although you need only one ServerInit and one ServerTerm directive, several Service directives often are needed, depending on how many different context roots are used.

9.3.2 IBM HTTP Server powered by Apache

For more information about how to configure the WebSphere Application Server plug-in into an IBM HTTP Server powered by Apache, see this website:

<https://ibm.biz/BdrNG9>

Two steps are required to configure the WebSphere Application Server plug-in into an IBM HTTP Server powered by Apache.

The first step is to make the WebSphere Application Server plug-in product code available to IBM HTTP Server powered by Apache. An example of the commands that are used to make the product code available is shown in Example 9-2 on page 141.

Example 9-2 Making the WebSphere Application Server plug-in code available in the server

```
cd /usr/lpp/zWebSphereEM1_Plugins/V8R5/bin
./install_plugin.sh -pluginInstallLocation /usr/lpp/zWebSphereEM1_Plugins/V8R5
-pluginRuntimeLocation /ihsconfig/ihs/ihsae002/Plugins -wasInstallLocation
/usr/lpp/zWebSphereEM85/V8R5
```

The second step is to configure the `httpd.conf` file of IBM HTTP Server powered by Apache to use the WebSphere Application Server plug-in. An example of the commands that are used to configure this file is shown in Example 9-3.

Example 9-3 Configuring the WebSphere Application Server plug-in into the httpd.conf file

```
cd /ihsconfig/ihs/ihsae002/Plugins/bin
./ConfigureIHSPlugin.sh -plugin.home /ihsconfig/ihs/ihsae001/Plugins
-plugin.config.xml /ihsconfig/ihs/ihsae002/Plugins/config/ihsae002/plugin-cfg.xml
-ihs.conf.file /ihsconfig/ihs/ihsae002/conf/httpd.conf -operating.system ZOS
-WAS.webserver.name ihsae001 -WAS.host.name wtsc55.itso.ibm.com
```

After the command that is shown in Example 9-3 completes, the last two lines of the `httpd.conf` file contain content similar to the excerpt that is shown in Example 9-4.

Example 9-4 An httpd.conf file excerpt

```
LoadModule was_ap22_module
/ihsconfig/ihs/ihsae002/Plugins/bin/mod_was_ap22_http.so
WebSpherePluginConfig
/ihsconfig/ihs/ihsae002/Plugins/config/ihsae002/plugin-cfg.xml
```

9.3.3 Key difference

How the two IBM HTTP Servers interact with the WebSphere Application Server plug-in module includes the following differences:

- ▶ In IBM HTTP Server powered by Domino, you must define a corresponding `Service` directive for any requests that you want to be processed by the WebSphere Application Server plug-in.
- ▶ In IBM HTTP Server powered by Apache, the plug-in is called first for any request that is received to see whether it is to be processed by the plug-in. If the plug-in finds that it does not process the request, it returns control so that the server can process it.

There is no equivalent Apache directive to the `Services` directive that is used in IBM HTTP Server powered by Domino because it is not required.

9.3.4 Working with the plug-in configuration file

The plug-in configuration file (`plugin-cfg.xml`) that is shown in Example 9-5 on page 142 includes routing information for all applications that are mapped to the web server. This file is read by the binary plug-in module that is loaded in the web server. An example of a binary plug-in module is the `mod_ibm_app_server_http.dll` file for IBM HTTP Server powered by Apache on the z/OS platform.

Example 9-5 An excerpt from the plugin-cfg.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?><!--HTTP server plugin config file for
the webserver ITS0Cell.wan.webserver1 generated on 2013.06.04 at 23:11:13 PM
BST-->
<Config ASDisableNagle="false" AcceptAllContent="false"
AppServerPortPreference="HostHeader" ChunkedResponse="false" FIPSEnable="false"
FailoverToNext="false" HTTPMaxHeaders="300" IISDisableNagle="false"
IISPluginPriority="High" IgnoreDNSFailures="false"
OS400ConvertQueryStringToJobsCCSID="false" RefreshInterval="60"
ResponseChunkSize="64" SSLConsolidate="true" TrustedProxyEnable="false"
VHostMatchingCompat="false">
  <Log LogLevel="Error"
Name="/opt/WebSphere/Plugins/logs/webserver1/http_plugin.log"/>
  <Property Name="ESIEnable" Value="true"/>
  <Property Name="ESIMaxCacheSize" Value="1024"/>
  <Property Name="ESIInvalidationMonitor" Value="false"/>
  <Property Name="ESIEnableToPassCookies" Value="false"/>
  <Property Name="PluginInstallRoot" Value="/IBMIHS/webserver1/Plugins*"/>
<VirtualHostGroup Name="default_host">
  <VirtualHost Name="*:9080"/>
  <VirtualHost Name="*:80"/>
  <VirtualHost Name="*:9443"/>
</VirtualHostGroup>

  <ServerCluster CloneSeparatorChange="false"GetDWLMTable="false"
IgnoreAffinityRequests="true" LoadBalance="Round Robin"
Name="server1_NodeA_Cluster" PostBufferSize="64" PostSizeLimit="-1"
RemoveSpecialHeaders="true" RetryInterval="60">
  <Server ConnectTimeout="0" ExtendedHandshake="false" MaxConnections="-1"
Name="NodeA_server1" WaitForContinue="false">
    <Transport Hostname="wan" Port="9080" Protocol="http"/>
    <Transport Hostname="wan" Port="9443" Protocol="https">
      <Property Name="keyring"
Value="/IBMIHS/webserver1/Plugins/config/webserver1/plugin-key.kdb"/>
      <Property Name="stashfile"
Value="/IBMIHS/webserver1/Plugins/config/webserver1/plugin-key.sth"/>
    </Transport>
  </Server>
</ServerCluster>

  <UriGroup Name="default_host_server1_NodeA_Cluster_URIs">
    <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid"
Name="/snoop*/"/>
    <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid"
Name="/hello"/>
  </UriGroup>
  <Route ServerCluster="server1_NodeA_Cluster"
UriGroup="default_host_server1_NodeA_Cluster_URIs"
VirtualHostGroup="default_host"/>
</Config>
```

The binary plug-in module does not change. However, the plug-in configuration file for the binary module must be regenerated and propagated to the web server whenever a change is made to the configuration of applications that are mapped to the web server. The binary module reads the XML file to adjust settings and to locate deployed applications for the web server.

The specific values for the `UriGroup Name` and `AffinityCookie` attributes depend on how you assembled your application. Consider the following when you assemble your application:

- ▶ If you specify `File Serving Enabled`, only a wildcard URI is generated, regardless of any explicit servlet mappings.
- ▶ If you specify `Serve servlets by class name`, a URI of the form `URI name = <webappuri>/servlet/` is generated.

Both of these options apply for the `Name` and `AffinityCookie` attributes.

When the plug-in configuration file is generated, it does not include `admin_host` in the list of virtual hosts. For more information about how to add it to the list, see the following IBM Knowledge Center website:

<https://ibm.biz/BdrNgH>

9.3.5 Regenerating the plug-in configuration file

The plug-in configuration file must be regenerated and propagated to the web servers when changes are made to your WebSphere configuration that affect how requests are routed from the web server to the application server. These changes include the following tasks:

- ▶ Installing an application
- ▶ Creating or changing a virtual host
- ▶ Creating a server
- ▶ Modifying HTTP transport settings
- ▶ Creating or altering a cluster

The plug-in file can be regenerated manually by using the administration tools. You can also set up the plug-in properties of the web server to enable automatic generation of the file whenever a relevant configuration change is made.

9.3.6 Managing who serves application static files

Typically, an application that is deployed in WebSphere Application Server consists of the application programs, such as servlets, EJBs, and static files, such as images and HTML. The default result is that requests for this static content mean that the WebSphere plug-in in IBM HTTP Server powered by Apache must fetch them. This fetch can be inefficient because it is more appropriate to have the V8.5.5 Server fetch the static content of the application. For more information, see this website:

<http://www.ibm.com/support/docview.wss?uid=swg21508890>

This website includes a description of an approach to have the IBM HTTP Server powered by Apache serve the static files of the application rather than the WebSphere Application Server.

A counter argument to whether you must use this approach is if your static content is static and your users are using a browser, the browser cached this static content and the WebSphere Application Server plug-in returns an HTTP response code of 304. The result is that your WebSphere Application Server does not have to serve that much static content.



Cache configuration

This chapter describes issues that are related to cache settings on IBM HTTP Server powered by Apache. These configurations and settings are used to enable cache in IHS powered by Apache environments only because this version does not support FRCA. This overview is specific to the z/OS operating system.

This chapter includes the following topics:

- ▶ 10.1, “Caching overview” on page 146
- ▶ 10.2, “Fast Response Cache Accelerator” on page 151

10.1 Caching overview

In IBM HTTP Server powered by Apache, `mod_cache` and `mod_file_cache` became standard production functions. These caching architectures provide a powerful means to accelerate HTTP handling, as an origin webserver and as a proxy.

The `mod_cache` and its provider modules (`mod_mem_cache` and `mod_disk_cache`) provide intelligent, HTTP-aware caching. Generally, use `mod_disk_cache` rather than `mod_mem_cache`. Caching static files is not recommended.

The content is stored in the cache, and `mod_cache` aims to honor all of the various HTTP headers and options that control the cacheability of content. It can handle local and proxied content. Also, `mod_cache` is aimed at simple and complex caching configurations in which you are dealing with proxied content, dynamic local content, or must speed up access to local files that change with time.

The `mod_file_cache` module presents a more basic form of caching. Rather than maintain the complexity of actively ensuring the cacheability of URLs, `mod_file_cache` offers file-handle and memory-mapping tricks to keep a cache of files as they were when V8.5.5 was last started. As such, `mod_file_cache` is aimed at improving the access time to local static files, which do not change often. Although `mod_file_cache` might be of use, as it was deprecated in Apache. We recommend that it is not used.

As `mod_file_cache` presents a relatively simple caching implementation (apart from the specific sections about `CacheFile` and `MMapFile`), the explanations in this publication cover the `mod_cache` caching architecture.

10.1.1 What can be cached

The two styles of caching in V8.5.5 work differently. The `mod_file_cache` caching maintains file contents as they were when V8.5.5 was started. When a request is made for a file that is cached by this module, it is intercepted and the cached file is served.

However, `mod_cache` caching is more complex. When serving a request, the caching module determines whether the content is cacheable if it was not cached previously. Determining cacheability of a response depends on the following conditions:

- ▶ Enabling and disabling caching for URLs is controlled by using the `CacheEnable` and `CacheDisable` directives.
- ▶ The response must have an HTTP status code of 200, 203, 300, 301, or 410.
- ▶ The request must be an HTTP GET request.
- ▶ If the request contains an “Authorization:” header, the response is not cached.
- ▶ If the response contains an “Authorization:” header, it must also contain an “s-maxage”, “must-revalidate”, or “public” option in the “Cache-Control:” header.
- ▶ If the URL included a query string (such as from an HTML form GET method), it is not cached unless the response specifies an explicit expiration by including an “Expires:” header or the *max-age* or *s-maxage* directive of the “Cache-Control:” header.
- ▶ If the response has a status of 200 (OK), the response must also include at least one of the “Etag”, “Last-Modified” or the “Expires” headers, or the *max-age* or *s-maxage* directive of the “Cache-Control:” header, unless the `CacheIgnoreNoLastMod` directive was used to require otherwise.

- ▶ If the response includes the “private” option in a “Cache-Control:” header, it is not stored unless the `CacheStorePrivate` was used to require otherwise.
- ▶ If the response includes the “no-store” option in a “Cache-Control:” header, it is not stored unless the `CacheStoreNoStore` was used.
- ▶ A response is not stored if it includes a “Vary:” header containing the match-all “*”.

10.1.2 Not cached

Any content that is highly time-sensitive or varies depending on the particulars of the request that are not covered by HTTP negotiation must not be cached.

If you have dynamic content that changes depending on the IP address of the requester or changes every 5 minutes, it must not be cached.

If the content that is served differs depending on the values of various HTTP headers, it might be possible to cache it intelligently by using a Vary header.

Variable and negotiated content

If a response with a Vary header is received by `mod_cache` when it is requesting content by the back-end, `mod_cache` attempts to handle the response intelligently. If possible, `mod_cache` detects the headers attributed in the Vary response in future requests and serves the correct cached response.

For example, if a response is received with a Vary header (as shown in Example 10-1), `mod_cache` serves only the cached content to requesters with `accept-language` and `accept-charset` headers matching the headers of the original request.

Example 10-1 Variable negotiate

```
Vary: negotiate,accept-language,accept-charset
```

10.1.3 File-handle caching

The act of opening a file can be a source of delay, particularly on network file systems. By maintaining a cache of open file descriptors for commonly served files, V8.5.5 can avoid this delay. Currently, V8.5.5 provides two different implementations of File-handle caching.

CacheFile

The most basic form of caching in V8.5.5 is the file-handle caching that is provided by `mod_file_cache`. Rather than caching file-contents, this cache maintains a table of open file descriptors. Files to be cached in this manner are specified in the configuration file by using the `CacheFile` directive. The `CacheFile` directive instructs V8.5.5 to open the file when V8.5.5 is started and to reuse this file-handle for all access to this file, as shown in Example 10-2.

Example 10-2 Cache file path example

```
CacheFile /usr/IBM/HTTPServer/htdocs/index.html
```

If you intend to cache many files in this manner, you must ensure that your operating system’s limit for the number of open files is set correctly. Although the use of `CacheFile` does not cause the file-contents to be cached, it does mean that changes that are made to the file while V8.5.5 is running are not picked up. The file is consistently served as it was when V8.5.5 was started.

If the file is removed while V8.5.5 is running, V8.5.5 continues to maintain an open file descriptor and serve the file as it was when V8.5.5 was started. Although the file was deleted and does not show up on the file system, extra free space is not recovered until V8.5.5 is stopped and the file descriptor is closed.

CacheEnable fd

The `mod_mem_cache` also provides its own file-handle caching scheme, which can be enabled by using the `CacheEnable` directive, as shown in Example 10-3.

Example 10-3 File-handle caching example

```
CacheEnable fd /
```

As with all of `mod_cache`, this type of file-handle caching is intelligent and handles are not maintained beyond the expiry time of the cached content.

10.1.4 In-memory caching

Serving directly from system memory is universally the fastest method of serving content. Reading files from a disk controller or even worse, from a remote network is orders of magnitude slower. Disk controllers usually involve physical processes, and network access is limited by your available bandwidth. However, memory access can take mere nanoseconds.

System memory is expensive. Byte for byte, it is the most expensive type of storage, so it is important to ensure that it is used efficiently. By caching files in memory, you decrease the amount of available memory on the system. In the case of operating system caching, this amount of memory is not so much of an issue. When V8.5.5's own in-memory caching is used, it is important to ensure that you do not allocate too much memory to a cache. Otherwise, the system is forced to swap out memory, which likely degrades performance.

Operating system caching

Almost all modern operating systems cache file-data in memory that is managed directly by the kernel. For example, we look at the difference in the time that it takes to read a file for the first time and the second time on z/OS and Linux, as shown in Example 10-4.

Example 10-4 Operating system caching

```
SC55@wsc55.itso.ibm.com:$ time cat testfile > /dev/null
real    0m0.065s
user    0m0.000s
sys      0m0.001s
SC55@wsc55.itso.ibm.com$ time cat testfile > /dev/null
real    0m0.003s
user    0m0.003s
sys      0m0.000s
```

Even for this small file, there is a huge difference in the amount of time it takes to read the file. This difference occurs because the kernel cached the file contents in memory.

By allocating spare memory on your system, you can ensure that more file-contents are stored in this cache. This method can be an efficient means of in-memory caching and involves no extra configuration of V8.5.5.

Because the operating system knows when files are deleted or modified, it can automatically remove file contents from the cache, when necessary. This feature is a significant advantage over V8.5.5's in-memory caching, which has no way of knowing when a file changes.

Despite the performance and advantages of automatic operating system caching, some circumstances occur in which in-memory caching might be better performed by V8.5.5. For example, an operating system can cache files only of which it is aware. If you are running V8.5.5 as a proxy server, the files you are caching are not locally stored but remotely served. If you still want the unbeatable speed of in-memory caching, V8.5.5's own memory caching is needed.

MMapFile caching

The MMapFile directive is provided by `mod_file_cache`. You can use this directive so that V8.5.5 maps a static file's contents into memory at start time (by using the `mmap` system call). V8.5.5 uses the in-memory contents for all accesses to this file, as shown in Example 10-5.

Example 10-5 MMapFile caching

```
MMapFile /usr/IBM/HTTPServer/htdocs/index.html
```

As with the CacheFile directive, any changes in these files are not picked up by V8.5.5 after it starts.

Overusing the directive: The MMapFile directive does not track how much memory it allocates; therefore, you must ensure that the directive is not overused. Each V8.5.5 child process replicates this memory, so it is critically important to ensure that the mapped files are not so large as to cause the system to swap memory.

mod_mem_cache caching

An HTTP-aware intelligent in-memory cache is provided by `mod_mem_cache`. It also uses heap memory directly, which means that even if MMap is not supported on your system, `mod_mem_cache` might still perform caching. Caching of this type is enabled, as shown in Example 10-6.

Example 10-6 Mod_mem_cache example

```
# Enable memory caching
CacheEnable mem /
# Limit the size of the cache to 1 Megabyte
MCacheSize 1024
```

10.1.5 Disk-based caching

A disk-based caching mechanism for `mod_cache` is provided by `mod_disk_cache`. As with `mod_mem_cache`, this cache is intelligent. Content is served from the cache only if it is considered valid.

Typically, the module is configured as shown in Example 10-7.

Example 10-7 Configuring `mod_disk_cache`

```
CacheRoot      /var/cache/IBM/HTTPServer/  
CacheEnable disk /  
CacheDirLevels 2  
CacheDirLength 1
```

Importantly, because the cached files are locally stored, operating system in-memory caching often also is applied to their access. Although the files are stored on disk, it is likely that the operating system ensures that they are served from memory if they are frequently accessed.

Understanding the cache-store

To store items in the cache, `mod_disk_cache` creates a 22 character hash of the URL that is requested. This hash incorporates the host name, protocol, port, path, and any CGI arguments to the URL to ensure that multiple URLs do not collide. Each character might be any one of 64-different characters, which means that overall there are 64^{22} possible hashes.

For example, a URL might be hashed to `xyTGxSM02b68mBCykqkp1w`. This hash is used as a prefix for the naming of the files that are specific to that URL within the cache. However, first it is split up into directories according to the `CacheDirLevels` and `CacheDirLength` directives.

`CacheDirLevels` specifies how many levels of subdirectory are created, and `CacheDirLength` specifies how many characters are in each directory. By using our example settings, the hash is turned into a file name prefix as `/var/cache/IBM/HTTPServer/x/y/TGxSM02b68mBCykqkp1w`.

The overall aim of this technique is to reduce the number of subdirectories or files that might be in a specific directory because most file systems slow down as this number increases. With setting of 1 for `CacheDirLength`, there can be (at most) 64 subdirectories at any specific level. With a setting of 2, there can be $64 * 64$ subdirectories, and so on. It is recommended to use a value of 1 for `CacheDirLength`.

Setting `CacheDirLevels` depends on how many files you anticipate to store in the cache. With the setting of 2 used in our example, a total of 4096 subdirectories can be created. With 1 million files cached, this figure works out at approximately 245 cached URLs per directory.

Each URL uses at least two files in the cache-store. Typically, there is a `.header` file, which includes meta-information about the URL, such as when it is due to expire. It also includes a `.data` file that is a copy of the content to be served.

In the case of content that is negotiated by using the Vary header, a `.vary` directory is created for the URL in question. This directory has many `.data` files corresponding to the differently negotiated content.

Maintaining the disk cache

Although `mod_disk_cache` removes cached content as files expire, it does not maintain any information about the total size of the cache or how little free space might be left.

Instead, the `htcacheclean` tool is provided with V8.5.5. As the name suggests, this tool is used to clean the cache periodically. Determining how frequently to run `htcacheclean` and what target size to use for the cache is complex; therefore, several attempts at using the tool might be needed to find the optimal values.

The `htcacheclean` tool has two modes: It can be run as a persistent daemon or periodically from cron. The `htcacheclean` tool can take up 1 hour or more to process large (tens of gigabytes) caches. If you are running the tool from cron, you must determine how long a typical run takes so that multiple instances are not run at the same time. Cache usage x interval between `htcacheclean` is shown in Figure 10-1.

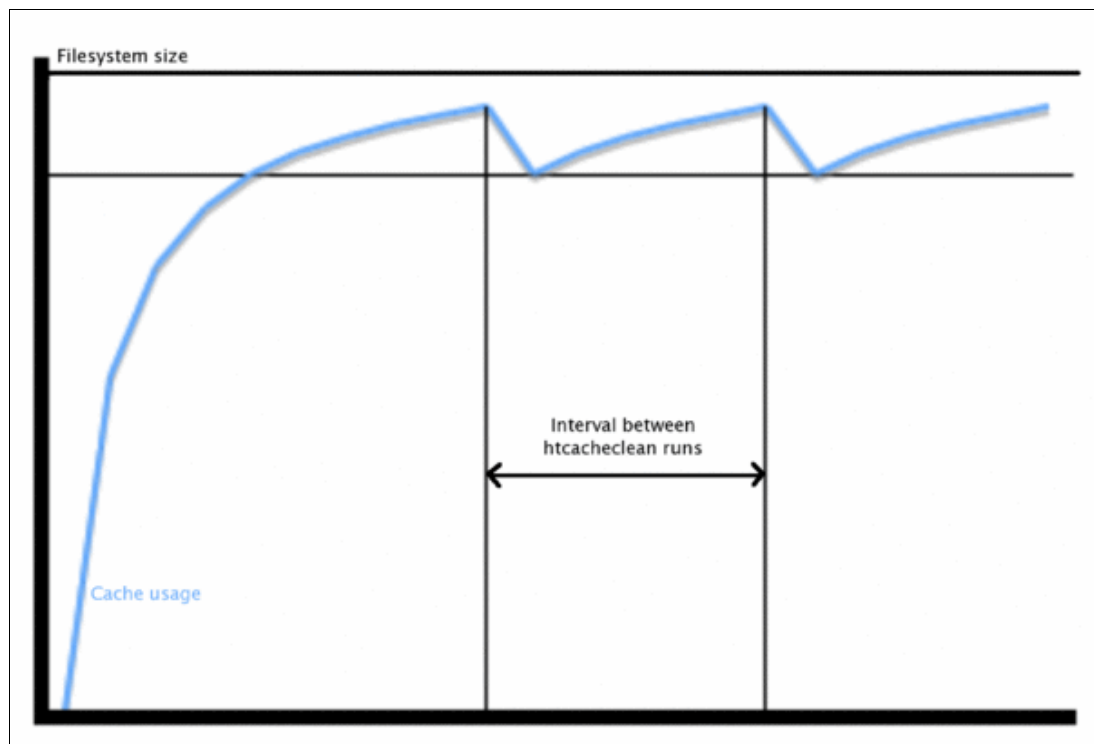


Figure 10-1 Typical cache growth and clean sequence

Because `mod_disk_cache` does not track how much space is used, you must ensure that `htcacheclean` is configured to leave enough room for growth following a clean.

For more information about static pages, see 9.3.6, “Managing who serves application static files” on page 143. For information about dynamic webpages and CGI, see Chapter 12, “CGI scripts” on page 163.

10.2 Fast Response Cache Accelerator

IBM HTTP Server powered by Domino supported Fast Response Cache Accelerator (FRCA). However, FRCA did not support requests that are received over SSL connections, which limited its usefulness.

FRCA is not supported in IBM HTTP Server powered by Apache.

Do not use in-memory or on-disk caching for any static content. Use caching for generated, proxied, or dynamic content.



Modules

This chapter describes the reasons why you might want to implement custom modules in your environment. It also describes briefly how DGW supported custom modules. Three sample Apache-style modules also are presented.

This chapter includes the following topics:

- ▶ 11.1, “Why custom modules are used” on page 154
- ▶ 11.2, “DGW modules” on page 154
- ▶ 11.3, “Simple helloworld module” on page 155
- ▶ 11.4, “Apache-supplied example module” on page 158
- ▶ 11.5, “Using an open source Apache module” on page 160

11.1 Why custom modules are used

DGW and V8.5.5 provide many features that you expect to find in an HTTP Server. However, no matter how many features of a product are provided, users sometimes have a business requirement that cannot be met by the product.

Although clients can request that a feature is added to the product, that process often takes time. The ability of DGW and V8.5.5 to allow clients to implement custom modules that add required capability in a short time is of great benefit.

11.1.1 Popularity of Apache modules

IHS is based on Apache, which means that you have access to many open source modules. Typically, these modules are run in Apache HTTP Servers on distributed platforms. By using V8.5.5 instead of DGW, you have access to a far greater range of developed modules. If you encounter problems developing your own modules or implementing a module, various Apache forums are available where you can request advice and assistance.

Taking an Apache module and implementing it in V8.5.5. DGW cannot compete with V8.5.5 in this regard, as described in 11.5, “Using an open source Apache module” on page 160.

If you must develop your own Apache modules, search for a book about this topic on the Internet. In addition, the following IBM Techdocs show examples of developing Apache modules that might also be of use in helping you to gain an understanding of this task:

- ▶ Extending the IBM HTTP Server for z/OS Powered by Apache with custom modules:
<http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101225>
- ▶ Classify URL requests in Apache IHS using WLM on z/OS:
<http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101858>

11.2 DGW modules

DGW allows you to develop custom modules by using the Go Webserver Application Programming Interface (GWAPI). The GWAPI modules are written in C code. Although GWAPI modules also can be written in Rexx, these modules are limited to specific exit routines. For more information about the use of GWAPI, see this website:

<https://ibm.biz/Bdr7Xq>

11.2.1 Migrating GWAPI modules to V8.5.5 modules

No utility is available that converts GWAPI modules to V8.5.5 modules. DGW and V8.5.5 (logically at a high level) provide a way to get the server to perform some action that they cannot perform by default. How you must code the modules to perform this migration is different for the two products.

The following sections feature sample Apache modules that can assist you with the task of converting DGW GWAPI modules to Apache-style modules.

11.3 Simple helloworld module

Apache is open source software that was developed by volunteers and is widely used. However, detailed documentation about working with custom modules in Apache-powered is not readily available. If you are new to the area of custom modules, it can prove challenging to get a custom module to work.

The purpose of the helloworld type custom module is to show the following information:

- ▶ Basics of how to code a custom module
- ▶ How to compile it
- ▶ How to integrate it into the HTTP Server
- ▶ How to test it

11.3.1 Code structure of helloworld module

Describing the inner workings of Apache modules is beyond the scope of this IBM Redpaper publication. User who require more information are encouraged to read published books about the subject. In this publication, we describe the basic concepts by using the code in the helloworld module.

The code starts with the following include statements:

```
#include "httpd.h"
#include "http_config.h"
```

These statements feature bringing in core components that are necessary for use when Apache modules are written. You always need these lines in any modules that you write.

The next piece of code is used to provide the “glue” between the Apache HTTP Server and the custom module. The HTTP Server can then determine what functions in the module can be started and when to start them, as shown in Example 11-1.

Example 11-1 AP_MODULE_DECLARE_DATA example

```
module AP_MODULE_DECLARE_DATA modHelloWorld_module =
{
    /* Only one callback function is provided. Real
     * modules will need to declare callback functions for
     * server/directory configuration, configuration merging
     * and other tasks. */
    STANDARD20_MODULE_STUFF,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    modHelloWorld_register_hooks, /* callback for registering hooks */
};
```

When the HTTP Server starts, it calls this module. Notice the number of NULLs in our example. These NULLs represent places where you can register hooks that process the server configuration.

The key line in our example is the line that includes `modHelloWorld_register_hooks`. The call to the `modHelloWorld_module` results in `modHelloWorld_register_hooks` being called. The following example is the code for `modHelloWorld_register_hooks`:

```
static void modHelloWorld_register_hooks (apr_pool_t *p)
{
    ap_hook_handler(modHelloWorld_method_handler, NULL, NULL, APR_HOOK_LAST);
}
```

The code is telling the HTTP Server about what hooks are available to the server. In this case, we are registering one hook only. You see in later sample modules that many hooks can be registered.

The code is registering the `modHelloWorld_method_handler` hook, which includes the code that is shown in Example 11-2.

Example 11-2 Registering the modHelloWorld_method_handler hook

```
static int modHelloWorld_method_handler (request_rec *r)
{
    /* Send a message to stderr (apache redirects this to the error log)*/
    fprintf(stderr,"apache2_modHelloWorld: A request was made.\n");
    /* Return DECLINED so that the Apache core will keep looking for
     * other modules to handle this request. This effectively makes
     * this module completely transparent. */
    return DECLINED;
}
```

The code consists of only one line, which prints a message only to the `stderr` of the Apache server.

11.3.2 Compiling the helloworld module

Apache includes a Perl script that is named `apxs` that is used to compile custom-written modules. Add the `helloworld` module source code to a directory of your choice on your z/OS system, for example `/var/ihsMods/modHelloWorld`.

The `apxs` script is included as part of the V8.5.5 product. In our environment, the script was placed in the `/ihs/usr/lpp/IHSA/V8R5/bin/apxs` directory. However, you cannot successfully start the `apxs` script from this directory. You must run the `install_ihs` script in the `/ihs/usr/lpp/IHSA/V8R5/bin` directory to define a V8.5.5 environment at a nominated directory.

On the system that was used to develop the modules that are described in this chapter, the defined V8.5.5 was in the `/ihsconfig/ihs/ihsae001` directory.

From the `/ihsconfig/helloWorld` directory, we issued the following command:

```
/ihsconfig/ihs/ihsae001/bin/apxs -c mod_helloWorld.c
```

The output that is shown in Example 11-3 was produced.

Example 11-3 Command output

```
/ihsconfig/ihs/ihsae001/build/libtool --mode=compile cc -Wc,XPLINK,lp64,dll,expo
-Wl,XPLINK,lp64 -O3 -U_NO_PROTO -DPHREAD_ATTR_SETDETACHSTATE_ARG2_ADDR
-DPTHREAD_SETS_ERRN
O -DPHREAD_DETACH_ARG1_ADDR -DSIGPROC_MASK_SETS_THREAD_MASK -DTCP_NODELAY=1
-I/ihsconfig/ihs/ihsae001/include -I/ihsconfig/ihs/ihsae001/include
-I/ihsconfig/ihs/ihsae001/i
nclude -Wc,DLL,expo -c -o mod_helloWorld.lo mod_helloWorld.c && touch
mod_helloWorld.slo
libtool.exe: cc -Wc,DLL,EXPORTALL -Wc,XPLINK,lp64,dll,expo -Wl,XPLINK,lp64 -O3
-U_NO_PROTO -DPHREAD_ATTR_SETDETACHSTATE_ARG2_ADDR -DPHREAD_SETS_ERRNO
-DPTHREAD_DETACH_ARG1_AD
DR -DSIGPROC_MASK_SETS_THREAD_MASK -DTCP_NODELAY=1
-I/ihsconfig/ihs/ihsae001/include -I/ihsconfig/ihs/ihsae001/include
-I/ihsconfig/ihs/ihsae001/include -Wc,DLL,expo -c -o mod_
helloWorld.o mod_helloWorld.c
libtool.exe: echo timestamp >mod_helloWorld.lo
/ihsconfig/ihs/ihsae001/build/shlibtool --mode=link cc -Wc,XPLINK,lp64,dll,expo
-Wl,XPLINK,lp64 -O3 -U_NO_PROTO -DPHREAD_ATTR_SETDETACHSTATE_ARG2_ADDR
-DPTHREAD_SETS_ERRN
O -DPHREAD_DETACH_ARG1_ADDR -DSIGPROC_MASK_SETS_THREAD_MASK -DTCP_NODELAY=1 -o
mod_helloWorld.la -rpath /ihsconfig/ihs/ihsae001/modules -module -avoid-version
-L/ihsconf
ig/ihs/ihsae001/lib -export-dynamic
--core-dll=/ihsconfig/ihs/ihsae001/lib/apache_core.dll mod_helloWorld.lo
libtool.exe: ar -rs mod_helloWorld.a mod_helloWorld.o
ar: creating mod_helloWorld.a
libtool.exe: cc -Wl,DLL -o mod_helloWorld.so -Wc,XPLINK,lp64,dll,expo
-Wl,XPLINK,lp64 -O3 -U_NO_PROTO -DPHREAD_ATTR_SETDETACHSTATE_ARG2_ADDR
-DPTHREAD_SETS_ERRNO -DPHREAD_DET
ACH_ARG1_ADDR -DSIGPROC_MASK_SETS_THREAD_MASK -DTCP_NODELAY=1 mod_helloWorld.o
/ihsconfig/ihs/ihsae001/lib/apache_core.x /ihsconfig/ihs/ihsae001/lib/libapr-1.x
/ihsconfig/ihs/ihsae001/lib/libaprutil-1.x /ihsconfig/ihs/ihsae001/lib/apache_core.x
/ihsconfig/ihs/ihsae001/lib/libapr-1.x
/ihsconfig/ihs/ihsae001/lib/
libaprutil-1.x
```

The contents of the directory now includes the following lines:

```
-rw-rw-r-- 1 IHSAE001 IHSRB13      1916 Jun 18 20:50 mod_helloWorld.c
-rw-rw-rw- 1 IHSAE001 IHSRB13         0 Jun 18 20:52 mod_helloWorld.slo
-rw-rw-rw- 1 IHSAE001 IHSRB13     4400 Jun 18 20:52 mod_helloWorld.o
-rw-rw-rw- 1 IHSAE001 IHSRB13       10 Jun 18 20:52 mod_helloWorld.lo
-rw-rw-rw- 1 IHSAE001 IHSRB13     4654 Jun 18 20:52 mod_helloWorld.a
-rw-rw-rw- 1 IHSAE001 IHSRB13       80 Jun 18 20:52 mod_helloWorld.x
-rwxrwxrwx 1 IHSAE001 IHSRB13    86016 Jun 18 20:52 mod_helloWorld.so
-rw-rw-rw- 1 IHSAE001 IHSRB13       564 Jun 18 20:52 mod_helloWorld.la
```

11.3.3 Integrating the new helloworld module into the configuration file

Next, we updated the HTTP Server configuration file to enable the modHelloWorld module. On our system, the configuration file is in the /ihsconfig/ihs/ihsae001/conf/httpd.conf directory.

Several LoadModule directives are in the configuration file. The last of these directives are similar to the following directives:

```
LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
#LoadModule rewrite_module modules/mod_rewrite.so
#LoadModule deflate_module modules/mod_deflate.so
```

Add the following line after the last LoadModule directive:

```
LoadModule mod_helloworld_module /ihsconfig/helloWorld/mod_helloworld.so
```

This code is placed on a single line.

11.3.4 Testing the helloworld module

We were running our V8.5.5 as a started task. We stopped and restarted it. We then entered the following URL into a browser to access the server:

```
http://wtsc55oe.itso.ibm.com:8230
```

The V8.5.5 home page was displayed. In the stderr log file at /ihsconfig/ihs/ihsae001/logs/error_log, we the following line was included:

```
[Tue Jun 18 21:23:30 2013] [warn] This message from mod_helloworld
```

This result shows that our helloworld module was started because of the V8.5.5 processing request that was sent.

11.4 Apache-supplied example module

This section describes how to compile, configure, and test the standard sample example module that is included with V8.5.5.

11.4.1 Code structure overview

IHS product code contains a more extensive sample module that is named the example module. On our system, this source in the following directory:

```
/ihs/usr/lpp/IHSA/V8R5/example_module/mod_example.c
```

When you create your own V8.5.5 environment as a result of running the install_ihs script, you find a symbolic link that is named example_module that points to the directory that is used. On our system, this directory was /ihsconfig/ihs/ihsae00. This sample module specifies hooks that can be used with the Apache-powered server, as shown in Example 11-4 on page 159.

Example 11-4 Specifying hooks

```
static void x_register_hooks(apr_pool_t *p)
{
    ap_hook_pre_config(x_pre_config, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_post_config(x_post_config, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_open_logs(x_open_logs, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_child_init(x_child_init, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_handler(x_handler, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_quick_handler(x_quick_handler, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_pre_connection(x_pre_connection, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_process_connection(x_process_connection, NULL, NULL, APR_HOOK_MIDDLE);
    /* Ý1" post read_request handling */
    ap_hook_post_read_request(x_post_read_request, NULL, NULL,
                              APR_HOOK_MIDDLE);
    ap_hook_log_transaction(x_logger, NULL, NULL, APR_HOOK_MIDDLE);
    #if 0
        ap_hook_http_method(x_http_method, NULL, NULL, APR_HOOK_MIDDLE);
        ap_hook_default_port(x_default_port, NULL, NULL, APR_HOOK_MIDDLE);
    #endif
    ap_hook_translate_name(x_translate_handler, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_header_parser(x_header_parser_handler, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_check_user_id(x_check_user_id, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_fixups(x_fixer_upper, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_type_checker(x_type_checker, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_access_checker(x_access_checker, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_auth_checker(x_auth_checker, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_insert_filter(x_insert_filter, NULL, NULL, APR_HOOK_MIDDLE);
}
```

Most of the hooks that are shown in Example 11-4 perform no action in the sample module. (It is beyond the scope of this paper to explain them in depth.) Documentation about these hooks can be difficult to find. Obtain a good book on this subject if you plan to develop your own module.

11.4.2 Compiling the example module

We copied the example module source code to the directory `/ihsconfig/example_module`. We then issued the following command to compile it:

```
/ihsconfig/ihs/ihsae001/bin/apxs -c mod_example.c
```

11.4.3 Integrating the example_module into the server conf file

Next, we integrated the `example_module` in the configuration file by using the same process that was described for the `helloworld` module in 11.3, “Simple helloworld module” on page 155 by adding the following line:

```
LoadModule example_module /ihsconfig/example_module/mod_example.so
```

To verify that `example_module` is working, add directives to tell the HTTP Server when to start it. These directives are included by adding the following lines at the bottom of the configuration file:

```
<Location /example-info>
  SetHandler example-handler
</Location>
Example directive declared here: YES
```

We then stopped and started the server.

11.4.4 Testing the `example_module`

To test that the Apache-powered HTTP Server can start the various hooks in the `example_module`, we entered the following URL:

`http://wtsc55oe.itso.ibm.com:8230/example-info`

The output is shown in the browser.

The top of this output is shown in Example 11-5.

Example 11-5 Top of output

```
mod_example Module Content-Handler Output
Apache HTTP Server version: "IBM_HTTP_Server"
Server built: "May 23 2013 00:51:38"
```

The format for the callback trace is:

```
n.<routine-name>(<routine-data>)
[<applies-to>]
```

The bottom of this output was as follows:

```
7. x_handler()
[DIR()]
```

Environment for this call:

- Applies-to: DIR()
 - "Example" directive declared here: YES
 - "Example" inherited: NO
-

11.5 Using an open source Apache module

Because the Apache HTTP Server is widely used, many modules were developed by the worldwide community. This section describes how you can take one of these open source modules and implement it in V8.5.5. This description demonstrates that even though these open source modules are not typically developed for use on z/OS, they can be implemented them in V8.5.5.

11.5.1 Limit IP module

The Limit IP module provides a simple mechanism to limit how many open connections are allowed from a single TCP/IP address.

The source code is available at this website:

<http://dominia.org/djao/limitipconn2.html>

Note: We found that when we copied the `mod_limitipconn.c` to a directory in z/OS UNIX, it was not correctly formatted and all of the code was on one line. We opened the `mod_limitipconn.c` file in WordPad on our Windows 7 workstation, added one space, and saved the file, which corrected the formatting issue.

11.5.2 Compiling the module

We copied `mod_limitipconn.c` to the `/ihsconfig/limitIp` directory. We then compiled it by using the following command:

```
/ihsconfig/ihs/ihsae001/bin/apxs -c mod_limitipconn.c
```

11.5.3 Updating the `httpd.conf` file

We added the following line to describe how to compile, configure, and test the standard sample example module that is included with the HTTP Server:

```
LoadModule limitipconn_module /ihsconfig/limitIp/mod_limitipconn.so
```

We added the following lines near the bottom of the `httpd.conf` file:

```
# Set a server-wide limit of 2 simultaneous downloads per IP,  
# no matter what.  
MaxConnPerIP 2
```

11.5.4 Restarting and testing

We then restarted the server. To test if this module worked, we coded a Rexx program that is named `sleeper.rx` to which we can pass a parameter to specify how long the Rexx program enters a sleep state. We placed `sleeper.rx` in the `/ihsconfig/ihs/ihsae001/cgi-bin` directory.

We then issued the following URL from three different browser windows on our workstation:

```
http://wtsc55.itso.ibm.com:8230/cgi-bin/sleeper.rx?sleep=30
```

The first two requests were received by our V8.5.5 server and went into a sleep state. However, the browser received the following message when we sent the third request:

```
Service Temporarily Unavailable  
The server is temporarily unable to service your request due to maintenance  
downtime or capacity problems. Please try again later.
```

```
IBM_HTTP_Server at wtsc55.itso.ibm.com Port 8230
```

In the V8.5.5 access_log, we saw the following messages:

```
9.190.237.213 - - [18/Jun/2013:22:01:25 -0400] "GET /cgi-bin/sleeper.rx?sleep=4
HTTP/1.1" 503 396
9.190.237.213 - - [18/Jun/2013:22:01:14 -0400] "GET /cgi-bin/sleeper.rx?sleep=30
HTTP/1.1" 200 1811
9.190.237.213 - - [18/Jun/2013:22:01:18 -0400] "GET /cgi-bin/sleeper.rx?sleep=30
HTTP/1.1" 200 1811
```

These messages show that the Limit IP module worked as expected by allowing two connections from our workstation at address 9.190.237.213, but rejecting the third request.



CGI scripts

This chapter describes the use of common gateway interface (CGI) scripts in IBM HTTP Server powered by Apache (V8.5.5) and IBM HTTP Server powered by Domino (DGW).

This chapter includes the following topics:

- ▶ 12.1, “CGI overview” on page 164
- ▶ 12.2, “Rexx CGI programs in DGW” on page 165
- ▶ 12.3, “Rexx CGI programs in V8.5.5” on page 166
- ▶ 12.4, “Perl CGI programs in V8.5.5” on page 173
- ▶ 12.5, “PHP CGI programs in V9” on page 174
- ▶ 12.6, “PHP CGI programs in V8.5.5” on page 185
- ▶ 12.7, “Lua support” on page 186

12.1 CGI overview

CGI provides a standard method to allow HTTP Servers to start a program that processes the received request and generates a reply. CGI programs can be written in several languages, such as C, Perl, PHP, and Rexx.

12.1.1 Brief history

The CGI approach was developed in the early 1990s when the Internet was just starting to take off. The use of CGI allowed HTTP Servers to do more than serve static HTML pages. The use of CGI programs allowed customized HTML to be returned to the user.

12.1.2 CGI disadvantage

HTTP Servers (including V8.5.5 and DGW) manage CGI processing by creating a thread to run the CGI program that the received request wants to start. This approach is effective because if the new thread encounters a problem running the CGI, only in that thread fails instead of the entire HTTP Server.

However, if the HTTP Server was processing numerous requests that started CGI programs, the HTTP Server used many resources to manage creating and ending threads to run the CGI programs. In the 1990s when computers were not nearly as powerful as they are today, this approach to starting CGI programs did not support large numbers of users.

12.1.3 CGI alternatives

Various alternative approaches to CGI were developed over the years, including FastCGI and the use of Apache modules, such as `mod_php`. These approaches provide an alternative to the standard CGI approach of creating a thread for each request, so they are more efficient.

In the 2000s, many clients moved away from using CGI programs. They used products, such as WebSphere Application Server as the environment to run programs that perform complex programming tasks while using the HTTP Server as a front end.

These alternatives are beyond the scope of this IBM Redpaper publication.

12.1.4 A use for CGI

There can be a question regarding whether it is worthwhile to use CGI programs. The answer is probably not for those situations in which you are supporting tens of thousands of users (although you can use CGI programs if you so want).

More likely, you might find it useful to use CGI programs for small scale tasks that need a quick and inexpensive solution for a small user base.

12.2 Rexx CGI programs in DGW

DGW supports the use of Rexx for coding CGI programs. This section describes how to set up DGW to run a supplied sample.

12.2.1 DGW support for CGI programs

For more information about the use of CGI programs in DGW, see this website:

<https://ibm.biz/Bdr7z9>

12.2.2 Sample Rexx CGI program

A sample Rexx CGI program is available at the following z/OS 1.13 IBM Knowledge Center website:

<https://ibm.biz/Bdr7z3>

We pasted this sample code into a text file on our workstation and named it `example.rx`. We then created a directory on the z/OS LPAR that was named `/ihsconfig/dws/ihsde001/cgi-bin` and copied the `example.rx` file to this directory.

We found that to run this Rexx, we needed to edit the `example.rx` file that we transferred to the z/OS directory. We added a space after the last character on the last line and saved the change.

12.2.3 Using exec directive

To allow DGW to start our `example.rx` CGI program, we added the `exec` directive:

```
Exec /cgi-bin/example* /ihsconfig/dws/ihsde001/cgi-bin/example*
```

This directive tells DGW that any request that matches `/cgi-bin/example*` is to look in the `/ihsconfig/dws/ihsde001/cgi-bin` directory for a match.

We then restarted our DGW server after making this change.

12.2.4 Running the example.rx CGI

We then used the following URL to start the `example.rx` CGI:

```
http://wtsc55.itso.ibm.com:8231/cgi-bin/example.rx?var1=1&var2=2&var3=3&var4=4&var5=5&var6=6
```

Some of the output that was produced is shown in Figure 12-1.

Processing forms with CGI scripts

```
This is an example of a CGI script which uses the cgiutils and cgiparse functions to
create a valid HTTP header and parse forms data, respectively. The cgiutils program
is used to create a set of HTTP headers for the CGI script output.
The following is an example of using the cgiparse command to query the values of
specific fields parsed from the QUERY_STRING.
The format of the cgiparse command is cgiparse -value fieldname
.
FORM_var2='2'; FORM_var3='3'; FORM_var4='4'; FORM_var5='5'; FORM_var6='6'.
.
Value for variable 1 = 1
Value for variable 2 = 2
Value for variable 3 = 3
Value for variable 4 = 4
Value for variable 5 = 5
Value for variable 6 = 6

Number of unique fields = 6
```

Figure 12-1 Portion of the output from example.rx

12.3 Rexx CGI programs in V8.5.5

IHS supports the use of Rexx for coding CGI programs. This section describes how you change the example.rx that was running in DGW so that it runs in V8.5.5.

12.3.1 Default cgi-bin setup

By default, V8.5.5 is configured so that any program in the cgi-bin directory can be run. The default directives that are used are shown in Figure 12-2.

```
ScriptAlias /cgi-bin/ "/ihsconfig/ihs/ihsae002/cgi-bin/"

<Directory "/ihsconfig/ihs/ihsae002/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

Figure 12-2 Default directives that allow execution of CGI programs

12.3.2 Changing example.rx to enable it for V8.5.5

We copied example.rx to the /ihsconfig/ihs/ihsae002/cgi-bin directory. We then reviewed the code to determine what must be changed to get it work in V8.5.5.

The second line of example.rx includes the following code:

```
'cgiutils -status 200 -ct text/html'
```


This line uses a module that is named `cgiutils` and is supplied with DGW to create an HTTP response header to be sent back to the browser. This line cannot be used in V8.5.5. We replaced that line with the following lines:

```
say 'Content-type: text/html;charset=UTF-8'  
say ''
```

These lines are performing the same task as what the `cgiutils` module perform, in that they are creating an HTTP Response header.

Note: The second line (`say ''`) must be included because it separates the HTTP Response header from the body of the reply.

The following line also must be reviewed:

```
'cgiparse -form'
```

That line produced the following output:

```
FORM_var1='1'; FORM_var2='2'; FORM_var3='3'; FORM_var4='4'; FORM_var5='5';  
FORM_var6='6'
```

There is no direct equivalent V8.5.5 module for the DGW `cgiparse` command. However, because that command is getting the query data that is sent by the request, we can achieve much the same thing by using the following line:

```
say 'query string: ' ENVIRONMENT('QUERY_STRING')
```

The next part of the Rexx code that must be replaced is shown in Figure 12-3.

```
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'
```

Figure 12-3 Rexx code to be changed

We replaced that code with the code that is shown in Figure 12-4.

```
query_string = ENVIRONMENT('QUERY_STRING')
IF query_string <> '' THEN
DO
  in = query_string
  DO I = 1 BY 1 UNTIL in=''
    PARSE VAR in key.i='val.i'&' In
    if key.i='var1' then var1=val.i
    else if key.i='var2' then var2=val.i
    else if key.i='var3' then var3=val.i
    else if key.i='var4' then var4=val.i
    else if key.i='var5' then var5=val.i
    else if key.i='var6' then var6=val.i
    numvars = i
  END
END

say 'Value for variable 1 = <CODE>'var1
say '<BR></CODE> '
say 'Value for variable 2 = <CODE>'var2
say '<BR></CODE>'
say 'Value for variable 3 = <CODE>'var3
say '<BR></CODE>'
say 'Value for variable 4 = <CODE>'var4
say '<BR></CODE>'
say 'Value for variable 5 = <CODE>'var5
say '<BR></CODE>'
say 'Value for variable 6 = <CODE>'var6
```

Figure 12-4 Replacement code to display input data

Finally, the following lines must be replaced:

```
say 'Number of unique fields = <CODE>'numvars
'cgiparse -form -count'
```

We replaced those lines with the following line:

```
say 'Number of unique fields = <CODE>'numvars
```

The numvars variable was set in the code that is shown in Figure 12-4.

Modification results

We then used the following URL to start the modified example.rx CGI in our V8.5.5 server. The only difference in the URL was the port number:

```
http://wtsc55.itso.ibm.com:8235/cgi-bin/example.rx?var1=1&var2=2&var3=3&var4=4&var5=5&var6=6
```

The output was the same as shown in Figure 12-1 on page 166. The only difference was a result of the code that we used to replace the 'cgiparse -form' as this produced the following line of output:

```
var1=1&var2=2&var3=3&var4=4&var5=5&var6=6
```

12.3.3 Support for cgiutils and cgiparse in V8.5.5.2

By using APAR PI07665, IBM is providing versions of the `cgiutils` and `cgiparse` modules for the IBM HTTP Server powered by Apache. This APAR is included as part of V8.5.5.2 of the IBM HTTP Server powered by Apache. For more information about this APAR, see this website:

<http://www.ibm.com/support/docview.wss?uid=isg1PI07665>

With these new modules, you do need to replace the use of `cgiutils` and `cgiparse` in CGI programs in the way that is described in 12.3.2, “Changing `example.rx` to enable it for V8.5.5” on page 166. This process simplifies migration of CGI programs that use `cgiutils` and `cgiparse`.

Because we can obtain pre-release versions of these modules, we performed the tests that are described in this section to verify that the modules functioned as expected.

Copying `example.rx`

We copied the `example.rx` that we used with DGW to our server by copying the following line:

```
/ihsconfig/dws/ihsde001/cgi-bin/example.rx
```

to the following line:

```
/ihsconfig/ihs/ihsae002/cgi-bin/example.rx
```

Copying modules to bin subdirectory

We copied the new modules to the `bin` subdirectory of our server, which in our case was the following directory:

```
/ihsconfig/ihs/ihsae002/bin
```

We changed the permissions to 775 so that the modules can be run.

Updating `PATH` in `envvars`

To make the new modules available to CGI programs, we then edited the `envvars` file for our server, which was in the following directory:

```
/ihsconfig/ihs/ihsae001/bin/envvars
```

We added the following line:

```
export PATH=/ihsconfig/ihs/ihsae002/bin:$PATH
```

Restarting and testing

We then restarted our server and used the following URL to start the `example.rx` CGI in our IBM HTTP Server powered by Apache server:

```
http://wtsc55.itso.ibm.com:8235/cgi-bin/example.rx?var1=1&var2=2&var3=3&var4=4&var5=5&var6=6
```

The output was the same as shown in Figure 12-1 on page 166. This result verified that we can run the version of `example.rx` that ran in DGW in IBM HTTP Server powered by Apache with no changes.

12.3.4 Escaped characters

When a browser sends a URL, it replaces unsafe ASCII characters with a “%” followed by two hexadecimal digits. This process is often referred to as *escaping of characters*.

IBM HTTP Server powered by Domino provided a routine that is named `cgiparse` that you can use to manage receiving URLs that include escaped characters. Then, you can convert these escaped characters back to the original characters.

IBM HTTP Server powered by Apache does not provide a similar routine. The Apache approach is that it expects whatever language the CGI program is written in to handle this issue. Languages, such as Perl and PHP, provide built-in functions to handle this conversion.

Rexx does not provide such a function. To provide this function in Rexx, we used an open source Rexx routine that is available at this website:

<http://www.slac.stanford.edu/slac/www/tool/cgi-rexx/>

The usage statement that is shown in Example 12-1 is available at this website.

Example 12-1 Permission statement

Permission granted to use and modify this library so long as the copyright above is maintained, modifications are documented, and credit is given for any use of the library.

We used the `deweb` Rexx routine from the following website:

<http://www.slac.stanford.edu/slac/www/tool/cgi-rexx/deweb>

Disclaimer:

IBM DOES NOT WARRANT OR REPRESENT THAT THE CODE PROVIDED IS COMPLETE OR UP-TO-DATE. IBM DOES NOT WARRANT, REPRESENT OR IMPLY RELIABILITY, SERVICEABILITY OR FUNCTION OF THE CODE. IBM IS UNDER NO OBLIGATION TO UPDATE CONTENT NOR PROVIDE FURTHER SUPPORT.

ALL CODE IS PROVIDED “AS IS,” WITH NO WARRANTIES OR GUARANTEES WHATSOEVER. IBM EXPRESSLY DISCLAIMS TO THE FULLEST EXTENT PERMITTED BY LAW ALL EXPRESS, IMPLIED, STATUTORY AND OTHER WARRANTIES, GUARANTEES, OR REPRESENTATIONS, INCLUDING, WITHOUT LIMITATION, THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF PROPRIETARY AND INTELLECTUAL PROPERTY RIGHTS. YOU UNDERSTAND AND AGREE THAT YOU USE THESE MATERIALS, INFORMATION, PRODUCTS, SOFTWARE, PROGRAMS, AND SERVICES, AT YOUR OWN DISCRETION AND RISK AND THAT YOU WILL BE SOLELY RESPONSIBLE FOR ANY DAMAGES THAT MAY RESULT, INCLUDING LOSS OF DATA OR DAMAGE TO YOUR COMPUTER SYSTEM.

IN NO EVENT WILL IBM BE LIABLE TO ANY PARTY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES OF ANY TYPE WHATSOEVER RELATED TO OR ARISING FROM USE OF THE CODE FOUND HEREIN, WITHOUT LIMITATION, ANY LOST PROFITS, BUSINESS INTERRUPTION, LOST SAVINGS, LOSS OF PROGRAMS OR OTHER DATA, EVEN IF IBM IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS EXCLUSION AND WAIVER OF LIABILITY APPLIES TO ALL CAUSES OF ACTION, WHETHER BASED ON CONTRACT, WARRANTY, TORT OR ANY OTHER LEGAL THEORIES.

THIRD PARTY SOFTWARE IS LICENSED AND DISTRIBUTED TO YOU BY THE THIRD PARTY DISTRIBUTORS AND/OR RESPECTIVE COPYRIGHT AND OTHER RIGHT HOLDERS UNDER THEIR TERMS AND CONDITIONS. IBM MAKES NO EXPRESS OR IMPLIED WARRANTIES OR REPRESENTATIONS WITH RESPECT TO SUCH SOFTWARE AND PROVIDES NO INDEMNITY FOR SUCH SOFTWARE. IBM GRANTS NO EXPRESS OR IMPLIED PATENT OR OTHER LICENSE WITH RESPECT TO AND IS NOT LIABLE FOR ANY DAMAGES ARISING OUT OF THE USE OF SUCH SOFTWARE.

When run on z/OS, the Rexx code results in ASCII characters. To get EBCDIC characters, we found the lines that are shown in Example 12-2.

Example 12-2 Original deweb code

```
DO WHILE POS('%',String)\=0
  PARSE VAR String Pre'%'+1 Ch +2 In
  IF DATATYPE(Ch,'X') & LENGTH(Ch)=2 THEN
    Ch=X2C(Ch)
  ELSE DO; In=Ch||In; Ch='%'; END
```

Then, we changed the lines as shown in Example 12-3.

Example 12-3 Modified deweb code

```
xx = 1
DO WHILE POS('%',String)\=0
  PARSE VAR String Pre'%'+1 Ch +2 In
  IF DATATYPE(Ch,'X') & LENGTH(Ch)=2 THEN do
    INTERPRET ch_hex.xx "= '" || ch || "'x"
    ch = AETranslate('E' ch_hex.xx)
    xx = xx + 1
  end
  ELSE DO; In=Ch||In; Ch='%'; END
```

We then developed the AETranslate routine and added that routine to our Rexx CGI program, the code for which is shown in Example 12-4.

Example 12-4 Rexx routine to do character translation

```
AETranslate: PROCEDURE; parse arg xFlag string
/*****
/*      EBCDIC To ASCII & ASCII To EBCDIC Translate Tables      */
*****/
E='000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F'x||,
'202122232425262728292A2B2C2D2E2F303132333435363738393A3B3C3D3E3F'x||,
'404142434445464748494A4B4C4D4E4F505152535455565758595A5B5C5D5E5F'x||,
'606162636465666768696A6B6C6D6E6F707172737475767778797A7B7C7D7E7F'x||,
'808182838485868788898A8B8C8D8E8F909192939495969798999A9B9C9D9E9F'x||,
'A0A1A2A3A4A5A6A7A8A9AAABACADAFAFB0B1B2B3B4B5B6B7B8B9BABBBBCDBEBF'x||,
'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECFD0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x||,
'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEFF0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x

A='00010203CF09D37FD4D5C30B0C0D0E0F10111213C7B408C91819CCCD831DD21F'x||,
'81821C84860A171B89919295A2050607E0EE16E5D01EEA048AF6C6C21415C11A'x||,
'20A6E180EB909FE2AB8B9B2E3C282B7C26A9AA9CDBA599E3A89E21242A293B5E'x||,
'2D2FDFDC9ADDDE989DACBA2C255F3E3FD78894B0B1B2FCD6FB603A2340273D22'x||,
'F861626364656667686996A4F3AFAEC58C6A6B6C6D6E6F7071729787CE93F1FE'x||,
'C87E737475767778797AEFC0DA5BF2F9B5B6FDB7B8B9E6BBBCBD8DD9BF5DD8C4'x||,
'7B414243444546474849CBCABEE8ECED7D4A4B4C4D4E4F505152A1ADF5F4A38F'x||,
'5CE7535455565758595AA0858EE9E4D130313233343536373839B3F7F0FAA7FF'x

if xFlag = "A"
then convString = translate(string,A,E)
else convString = translate(string,E,A)
return convString
```

In the `example.rx`, we then modified the code as shown in Example 12-5.

Example 12-5 Handling escaped characters in query string

```
query_string = ENVIRONMENT('QUERY_STRING')
IF query_string <> '' THEN
DO
  in = query_string
  DO I = 1 BY 1 UNTIL in=''
    PARSE VAR in key.i='val.i'&' In
      key.i=DeWeb(key.i,"+")
      val.i=DeWeb(val.i,"+")
    if key.i='var1' then var1=val.i
    else if key.i='var2' then var2=val.i
    else if key.i='var3' then var3=val.i
    else if key.i='var4' then var4=val.i
    else if key.i='var5' then var5=val.i
    else if key.i='var6' then var6=val.i
    numvars = i
  END
END
```

We tested this code by using the following URL (the result is shown in Example 12-6):

`http://wtsc55.itso.ibm.com:8235/cgi-bin/example.rx?var1=reservedChars%24%26%2B%2c%2f%3a%3b%3d%3f%40&var2=unsafeChars%20%22%3c%3e%23%25%7b%7d%7c%5c%5e%5b%5d%60`

Example 12-6 Result of processing URL containing escaped characters

```
Value for variable 1 = reservedChars$&+,:;=?@
Value for variable 2 = unsafeChars "<>#%{}|\^~[]`"
```

This result showed that we can get our Rexx CGI program to correctly handle escaped characters.

12.3.5 Rexx CGI summary

We showed the steps that we used to modify a Rexx program that was running in DGW to get it to run in V8.5.5.

Although this program was a simple Rexx program, it does indicate that you can migrate Rexx programs that are running in DGW to V8.5.5 if needed. The main changes that you must make focus on how your Rexx code obtains information about the request.

12.3.6 More complex Rexx sample

For more information about an example of a more complex Rexx CGI program running in V8.5.5, see the IBM Techdoc *Using IBM HTTP Server and Rexx to view z/OS STC output using SDSF*, which is available at this website:

<http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD106087>

12.4 Perl CGI programs in V8.5.5

Perl is another language that can be used for CGI programs in V8.5.5.

12.4.1 Using Perl on z/OS

You must acquire and install Perl on z/OS to use the program. Perl previously was included as part of the IBM Ported Tools. For more information about Perl, see this website:

<http://www.ibm.com/systems/z/os/zos/features/unix/ported/perl/>

Rocket Software provides an Open Source port of Perl. For more information, see this website:

<http://www.rocketsoftware.com/ported-tools>

Verify with the provider the support that is available for the software if you want support.

12.4.2 Sample Perl CGI program

To show the use of Perl for CGI programs, we created a file in the `cgi-bin` directory of our V8.5.5 server that is named `helloWorld.perl`. The code for our sample Perl CGI program is shown in Figure 12-5.

```
#!/usr/bin/perl

print "Content-type: text/html\n\n";
print <<"EOF";
<html>
<head>
<title>IBM ITSO Simple Perl CGI</title>
</head>
<body>
<h1>A Simple Perl CGI from the ITSO</h1>
<p>Hello World</p>
</body>
</html>
EOF
```

Figure 12-5 Our sample Perl CGI program

12.4.3 IHS and LIBPATH

To run Perl programs, the `PATH` and `LIBPATH` must be configured to ensure that the Perl product code can be found.

In our `/usr/lib`, a symbolic link to the `libperl.so` module was included, as shown in Figure 12-6.

```
$ cd /usr/local/lib
$ ls -lrt | grep libperl
lrwxrwxrwx 1 OMVSKERN SYS1          57 Mar 12 2007 libperl.so ->
/shared/perl/lib/5.8.7/os390-thread-multi/CORE/libperl.so
```

Figure 12-6 Symlink to `libperl.so`

We must update the V8.5.5 setup so that it can run Perl CGI programs. This process was done by updating the `/ihsconfig/ihs/ihsae002/bin/envvars` file by adding the following line:

```
export LIBPATH=/usr/lib:$LIBPATH
```

We also added the following directive near the bottom of the `httpd.conf` file so that the updated LIBPATH is passed to the created processes in V8.5.5:


```
PassEnv LIBPATH
```

12.4.4 Testing the Perl CGI program

We used the following URL to start our Perl CGI program:

```
http://wtsc55.itso.ibm.com:8235/cgi-bin/helloWorld.perl
```

This produced the output shown in Figure 12-7.



```
A Simple Perl CGI from the ITSO  
Hello World
```

Figure 12-7 Output from running the Perl CGI program

12.5 PHP CGI programs in V9

This section describes an updated process of how to run PHP CGI programs in IBM HTTP Server powered by Apache V9. It reflects the change in how customers can obtain the PHP software from Rocket Software.

12.5.1 Using php on z/OS

In the first edition of this IBM Redpaper publication, we wrote that you can download the PHP for CGI programs by using the IBM Ported Tools site at this website:

<http://www.ibm.com/systems/z/os/zos/features/unix/ported/php/>

However, IBM no longer provides the PHP product.

Rocket Software provides an Open Source port of PHP, which is available at this website:

<http://www.rocketsoftware.com/ported-tools>

Verify with the provider the support that is available for the software if you want support.

12.5.2 Rocket PHP software

On our z/OS LPAR, the Rocket PHP software was installed in the `/u/rocket` directory.

12.5.3 Running PHP CGI programs

There are two ways you can configure PHP CGI programs to run in IBM HTTP Server powered by Apache.

Action approach

The first approach is to set IBM HTTP Server powered by Apache to start the PHP interpreter and pass to it the name of the PHP CGI program it runs. For more information about this approach, see 12.5.4, “PHP by using the action approach” on page 175.

Shebang approach

The second approach is to set IBM HTTP Server powered by Apache to start the PHP CGI program directly. For more information, see 12.5.5, “PHP by using the shebang approach” on page 181.

For more information about these approaches, see this website:

http://publib.boulder.ibm.com/httserv/ihsdiag/dgw_migration_faq.html#action

The description for PHP in 12.5.4, “PHP by using the action approach” and 12.5.5, “PHP by using the shebang approach” on page 181 also applies to other languages that are run as CGI, such as Perl.

12.5.4 PHP by using the action approach

Running PHP CGI programs by using the action approach requires that the first line of the PHP program does not contain a shebang line on the following form:

```
#!/u/rocket/bin/php-cgi
```

Instead, the first line uses the following format:

```
<?php
```

Sample PHP CGI program

Example 12-7 shows the PHP CGI program we wrote to demonstrate running PHP CGI programs. We added code to display the user ID that the PHP CGI program is running under. This display demonstrates how to set up security controls around running PHP CGI programs.

Example 12-7 The helloWorld.php program

```
<?php
$inipath = php_ini_loaded_file();
if ($inipath) {
    $iniPath = $inipath;
} else {
    $iniPath = 'A php.ini file is not loaded';
}
$phpPath = $_SERVER["SCRIPT_NAME"];
$break = Explode('/', $phpPath);
$file = $break[count($break) - 1];
$phpOwner = get_current_user();
$uid = posix_getuid();
$processUser = posix_getpwuid(posix_geteuid());
$username = getenv('USERNAME') ?: getenv('USER');
date_default_timezone_set('America/New_York');
$localTime = date("d/m/Y h:i:s a");
$ourPath = getcwd();
print <<<eot
<html>
```

```

<head>
<title>IBM ITS0 Simple php CGI </title>
</head>
<body>
<h1>A Simple php CGI from the ITS0 </h1>
<p>Hello from a php CGI program</p>
</body>
<p>This PHP running under the user ID: {$processUser}'name'"}</p>
<p>uid of the user ID: {$uid}</p>
<p>This PHP loaded from: {$ourPath}/{ $pfile}</p>
<p>PHP ini loaded from: {$iniPath}</p>
<p>PHP CGI program owner: {$phpOwner}</p>
<p>Run at time: {$localTime}</p>
</html>
eot;
?>

```

Directives in httpd.conf

We added the directives that are shown in Example 12-8 to the httpd.conf file to run PHP CGI programs without security.

Example 12-8 Directives added to httpd.conf file to run without security

```

Alias /noSecurePhp/ /ihsconfig/ihs/ihsae009/cgi-bin/phpNoSecure/

<Directory /ihsconfig/ihs/ihsae009/cgi-bin/phpNoSecure/>
AddHandler indirect-noSecurePhp-script .php
</Directory>

Action indirect-noSecurePhp-script /cgi-bin/noSecurePhp.sh

ScriptAliasMatch /cgi-bin/noSecurePhp.sh
/ihsconfig/ihs/ihsae009/cgi-bin/noSecurePhp.sh

<Location /cgi-bin/noSecurePhp.sh>
    Options +ExecCGI
    SetHandler cgi-script
</Location>

```

We added the directives that are shown in Example 12-9 to the httpd.conf file to run PHP CGI programs with security.

Example 12-9 Directives added to httpd.conf file to run with security

```

Alias /securePhp/ /ihsconfig/ihs/ihsae009/cgi-bin/phpSecure/

<Directory /ihsconfig/ihs/ihsae009/cgi-bin/phpSecure/>
AddHandler indirect-securePhp-script .php
</Directory>

Action indirect-securePhp-script /cgi-bin/securePhp.sh

ScriptAliasMatch /cgi-bin/securePhp.sh
/ihsconfig/ihs/ihsae009/cgi-bin/securePhp.sh

```

```

<Location /cgi-bin/securePhp.sh>
    Options +ExecCGI
    SetHandler cgi-script
    # Since our application runs entirely beneath the wrapper, it's where
    # we must configure things like SAFRunAs
    Authname "LOGON REQUIRED"
    AuthType Basic
    AuthBasicProvider saf
    Require valid-user
    SAFRunAs %%CLIENT%%
<RequireAll>
    Require valid-user
    # Prevent direct access to the wrapper
    Require env REDIRECT_STATUS
</RequireAll>
</Location>

```

The directives were annotated as shown in Figure 12-8 to help explain how IBM HTTP Server powered by Apache processes a request to run a PHP CGI program by using the action approach.

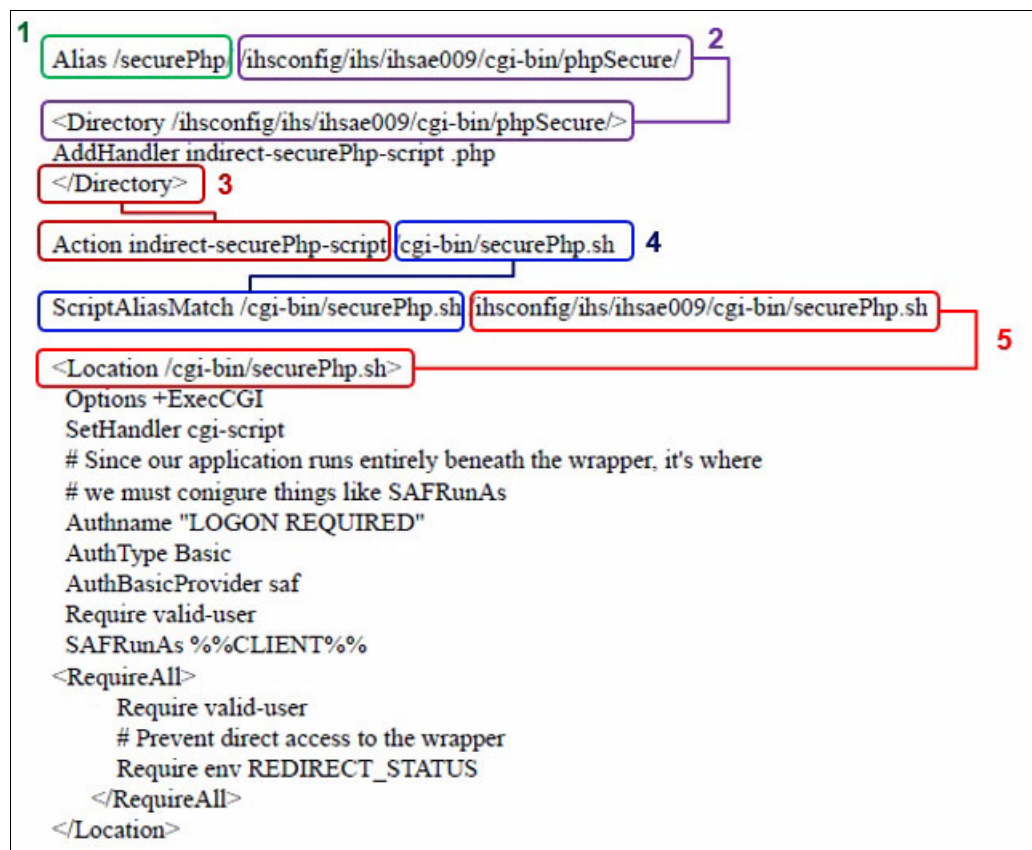


Figure 12-8 Interaction of directives

We sent the following URL:

<http://wtsc55.itso.ibm.com:8229/securePhp/helloWorld.php>

This URL is highlighted by “1” the Alias directive in Figure 12-8 on page 177.

The Alias directive indicates that this URL matches to the following directory in UNIX System Services:

```
/ihsconfig/ihs/ihsae009/cgi-bin/phpSecure
```

The following components are shown in Figure 12-8 on page 177:

- ▶ The directory block (highlighted by “2”) contains an AddHandler directive, which specifies the name of a handler that is named `indirect-securePhp-script` to be used for any file ending with “.php”.
- ▶ The Action directive (highlighted by “3”) specifies the name of a CGI script to be started for the specified handler, which in this case is `/cgi-bin/securePhp.sh`. This shell script is also referred to as a *wrapper script*.
- ▶ The ScriptAliasMatch directive (highlighted by “4”) associates the specified CGI wrapper script of `/cgi-bin/securePhp.sh` with a location in the UNIX System Services file system.
- ▶ The Location block (highlighted by “5”) is where we specify the security directives that are used to control access to run this PHP CGI program.

When the IBM HTTP Server powered by Apache runs the `securePhp.sh`, it passes the name of the PHP CGI program that is specified in the URL, which in this case is `securePhp/helloWorld.php`.

Wrapper CGI shell scripts

The wrapper CGI shell scripts `noSecurePhp.sh` and `securePhp.sh` include the following lines:

```
#!/bin/sh
/u/rocket/bin/php-cgi $SCRIPT_FILENAME
```

The `cgi.fix_pathinfo` option in `php.ini`

The PHP product features a default configuration file in a file that is named `php.ini`.

To use the action approach, the `php.ini` file must be changed from the following option:

```
;cgi.fix_pathinfo=1
```

to the following option:

```
cgi.fix_pathinfo=0
```

However, you must consider where to locate the `php.ini` file with this change so that it is found at run time.

On our z/OS LPAR, the PHP software was installed in the `/u/rocket` directory, which includes the supplied `php.ini` file.

For the directives, we used the action approach. At run time, IBM HTTP Server powered by Apache cannot find the `php.ini` file at this location.

The following options are available to correct this issue:

- ▶ Copy the supplied `php.ini` file to the same directory that the PHP wrapper shell script you created is in. In our case, we placed the two PHP wrapper scripts in the following directory:

```
/ihsconfig/ihs/ihsae001/cgi-bin
```

We also copied the supplied `php.ini` file to this directory and changed `cgi.fix_pathinfo` as described in this section.

- Supply the location of the modified `php.ini` file by using a parameter in the wrapper shell script. We tested this approach by changing the `noSecurePhp.sh` script, as shown in the following example:

```
#!/bin/sh
/u/rocket/bin/php-cgi --php-ini /u/edmcarr/php.ini $SCRIPT_FILENAME
```

If you attempt to run a PHP CGI program by using the action method and the PHP runtime environment does not find a `php.ini` file with `cgi.fix_pathinfo=0` specified, the following error message appears in the browser:

No input file specified.

Setting up php CGI subdirectory

To test the action approach, we included the following shell scripts in the `cgi-bin` subdirectory (see “Wrapper CGI shell scripts” on page 178):

```
noSecurePhp.sh
securePhp.sh
```

The following subdirectories were defined:

```
phpNoSecure
phpSecure
```

Into each of these subdirectories, we put a copy of our `helloWorld.php` CGI program.

Testing the action approach

We entered the following URL to access our PHP CGI program *without* security:

<http://wtsc55.itso.ibm.com:8229/noSecurePhp/helloWorld.php>

The output that is shown in Example 12-10 was received.

Example 12-10 Output from running using action approach without security

A Simple php CGI from the ITS0

Hello from a php CGI program

This PHP running under the user ID: IHSASTC

uid of the user ID: 35065

This PHP loaded from: /ihsconfig/ihs/ihsae009/cgi-bin/phpNoSecure/noSecurePhp.sh

PHP ini loaded from: /ihsconfig/ihs/ihsae009/cgi-bin/php.ini

PHP CGI program owner: IHSAE001

Run at time: 28/06/2016 12:59:51 am

You can see that when the PHP CGI program is run with no security, it ran under the user ID that the started task is running under.

We entered the following URL to access our PHP CGI program *with* security:

<http://wtsc55.itso.ibm.com:8229/securePhp/helloWorld.php>

We received this prompt for a user ID and password, as shown in Figure 12-9.

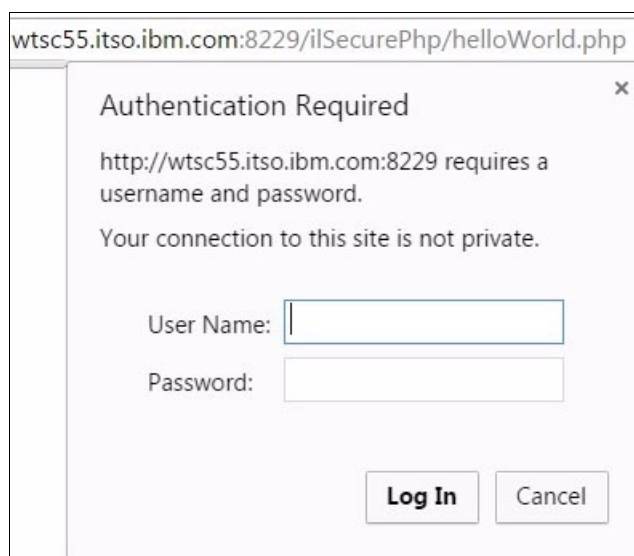


Figure 12-9 Browser prompting for user ID and password

After entering the user ID and password, we received the output that is shown in Example 12-11.

Example 12-11 Output from running using the action approach with security

Hello from a php CGI program

This PHP running under the user ID: EDMCAR

uid of the user ID: 2257

This PHP loaded from: /ihsconfig/ihs/ihsae009/cgi-bin/phpSecure/securePhp.sh

PHP ini loaded from: /ihsconfig/ihs/ihsae009/cgi-bin/php.ini

PHP CGI program owner: IHSAE001

Run at time: 27/06/2016 09:02:02 pm

You can see that when run with security, the PHP CGI program ran under the user ID that was used to authenticate, not the user ID that the started task is running under.

Preventing direct calling of the wrapper shell

When this approach is used, incorrect coding of the directives can result in an external user directly calling the wrapper shell script.

For example, it can allow the following URL to be processed:

`http://wtsc55.itso.ibm.com:8229/cgi-bin/securePhp.sh?/u/edmcar/helloWorld.php`

This URL is requesting the server to run the `securePhp.sh` wrapper script and trying to pass the name of a php CGI program to run.

To prevent this issue, you must code the following directive:

```
Require env REDIRECT_STATUS
```

With this directive coded (as described in “Directives in httpd.conf” on page 176), any attempt to start the wrapper shell script in a URL resulted in the message that is shown in Example 12-12 appearing in the browser.

Example 12-12 Output showing prevention of wrapper script being run

Forbidden

You don't have permission to access /cgi-bin/noSecurePhp.sh on this server.

When we sent the URL to start the PHP program where we did not code the `Require` directive, it did not start the specified CGI program that we tried to pass as a target program. However, the server did start the specified wrapper shell script.

The `Require env REDIRECT_STATUS` syntax of the `Require` directive is the Apache 2.4 directive on which IBM HTTP Server powered by Apache V9 is built. If you use IBM HTTP Server powered by Apache V8.5 (which is built on an older Apache 2.2), the following syntax is used:

```
allow from env=REDIRECT_STATUS
```

Spawned started tasks

For each PHP CGI program that is started by using the action approach, two started tasks are spawned to manage running the CGI program. These spawned started tasks feature names that are based on the parent started task name.

12.5.5 PHP by using the shebang approach

This approach runs the target CGI script through the interpreter that is specified on the first line of the CGI script. This first line is referred to as the *shebang line*.

Running PHP CGI programs by using the shebang approach requires that the first line of the PHP program consist of a line that is shown in the following example:

```
#!/u/rocket/bin/php-cgi
```

This line (the shebang line) must start with the `#!` characters and then specify the location of the PHP interpreter.

On our z/OS LPAR, the PHP software was installed under the `/u/rocket` directory.

Sample PHP CGI program

We used the same PHP CGI program that we used for the action approach, but added the shebang line as a first line.

We added the following line as the shebang first line to the sample PHP CGI program that we used in the testing of the action approach:

```
#!/u/rocket/bin/php-cgi
```

Directives in httpd.conf

We added the directives that are shown in Example 12-13 to the `httpd.conf` file to run PHP CGI programs *without* security.

Example 12-13 Directives added to run PHP CGI programs without security

```
Alias /ilNoSecurePhp/ /ihsconfig/ihs/ihsae009/cgi-bin/ilphpNoSecure/
<Directory /ihsconfig/ihs/ihsae009/cgi-bin/ilphpNoSecure/>
    Options +ExecCGI
    SetHandler cgi-script
</Directory>
```

We added the directives that are shown in Example 12-14 to the httpd.conf file to run PHP CGI programs *with* security.

Example 12-14 Directives added to run PHP CGI programs with security

```
Alias /ilSecurePhp/ /ihsconfig/ihs/ihsae009/cgi-bin/ilphpSecure/
<Directory /ihsconfig/ihs/ihsae009/cgi-bin/ilphpSecure/>
    Options +ExecCGI
    SetHandler cgi-script
    Authname "LOGON REQUIRED"
    AuthType Basic
    AuthBasicProvider saf
    Require valid-user
    SAFRunAs %%CLIENT%%
</Directory>
```

PHP CGI subdirectory setup

To test the action approach, we defined the following subdirectories in the cgi-bin subdirectory:

```
ilphpNoSecure
ilphpSecure
```

We included a copy of our helloWorld.php CGI program in each of these subdirectories.

Specify php.ini location

We found in our testing of the shebang approach, that the only way we could get our modified php.ini file to be used was to specify its location in the php shell program.

In each of the php programs we used to test the shebang approach we coded the first line in this fashion, using the -nc option to specify the location of our modified php.ini file:

```
#!/u/rocket/bin/php-cgi -nc/ihsconfig/ihs/ihsae099/cgi-bin/php.ini
```

The cgi.force_redirect option in php.ini

The use of the shebang approach requires a php.ini file with the cgi.fix_pathinfo set to 0 as it was for the action approach.

However, it also requires that another option in the php.ini file is modified.

To use the shebang approach, the following php.ini file option must be changed from ;cgi.force_redirect=1 to cgi.force_redirect=0.

Testing shebang PHP CGI program

We entered the following URL to access our PHP CGI program *without* security:

```
http://wtsc55.itso.ibm.com:8229/ilNoSecurePhp/helloWorld.php
```


The output that is shown in Example 12-15 on page 183 was received.

Example 12-15 Output from running using shebang approach with no security

Hello from a php CGI program

This PHP running under the user ID: IHSBESTC

uid of the user ID: 35065

This PHP loaded from: /ihsconfig/ihs/ihsae009/cgi-bin/ilphpNoSecure/helloWorld.php

PHP ini loaded from: /ihsconfig/ihs/ihsae009/cgi-bin/php.ini

PHP CGI program owner: IHSAE001

Run at time: 28/06/2016 01:20:08 am

We entered the following URL to access our PHP CGI program *with* security:

http://wtsc55.itso.ibm.com:8229/ilSecurePhp/helloWorld.php

We were prompted for a user ID and password, as shown in Figure 12-10.

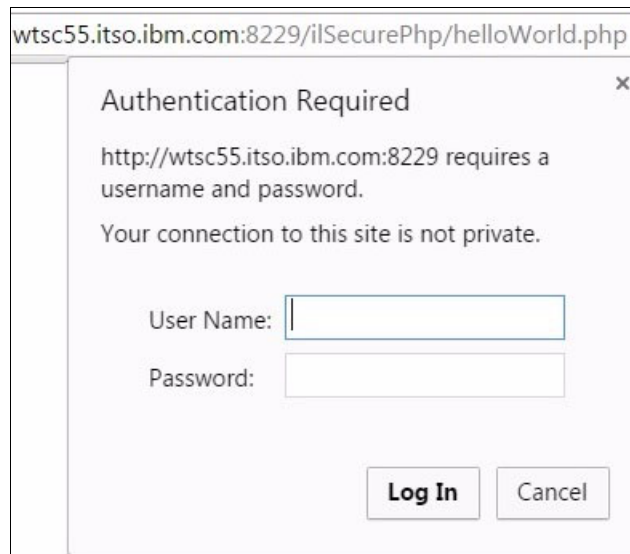


Figure 12-10 Browser prompting for user ID and password

After entering the user ID and password, we received the output that is shown in Example 12-16.

Example 12-16 Output from running shebang approach with security

A Simple php CGI from the ITS0

Hello from a php CGI program

This PHP running under the user ID: EDMCAR

uid of the user ID: 2257

This PHP loaded from: /ihsconfig/ihs/ihsae009/cgi-bin/ilphpSecure/helloWorld.php

PHP ini loaded from: /usr/local/php/lib/php.ini

PHP CGI program owner: IHS AE001

Run at time: 28/06/2016 01:22:08 am

As shown in Example 12-16 on page 183, the PHP CGI program ran under the EDMCAR user ID, which was the user ID we entered in the browser security prompt.

Specifying location of php.ini file

In the output that is shown in Example 12-15 on page 183 and Example 12-16 on page 183 that the line that starts with “PHP ini loaded from:” shows a different location for the php.ini file.

This difference occurs because we used the following code in the first line of the helloWorld.php program that was used to test the shebang approach without security:

```
#!/u/rocket/bin/php-cgi -nc/ihsconfig/ihs/ihsae099/cgi-bin/php.ini
```

The -nc option is used to specify the php.ini file that you want to be used by PHP when the CGI program is run.

In the helloWorld.php program that was used to test the shebang approach with security, the following code was used in the first line:

```
#!/u/rocket/bin/php-cgi
```

This code results in PHP loading the php.ini file from its default location of /usr/local/php/lib/php.ini.

During testing, we set the first line of our helloWorld.php to the following code:

```
#!/u/rocket/bin/php-cgi -nc/usr/local/php/lib/php.ini
```

This code specifies the default location of the php.ini file. When we ran our test, we saw the following output:

PHP ini loaded from: A php.ini file is not loaded

We were sure that the php.ini file at /usr/local/php/lib/php.ini was being used because the request failed if we changed the cgi.force_redirect to 1. However, we did not determine why the call to php_ini_loaded_file() in the helloWorld.php program returned no value when the first line specified the default php.ini location.

Spawned started task

For each PHP CGI program that is started by using the action approach, one started task is spawned to manage running the CGI program. The spawned started task includes a name that is based on the parent started task name.

12.6 PHP CGI programs in V8.5.5

This section is the original content that describes how to run a PHP CGI program by using IBM HTTP Server powered by Apache V8.5.5 with the PHP software that is included as part of the IBM Ported Tool software.

This content is presented here in the updated version of this IBM Redpaper publication in case this version is used.

The advice that is described in 12.5, “PHP CGI programs in V9” on page 174 also is applicable for use with IBM HTTP Server powered by Apache V8.5.5.

12.6.1 Sample php CGI program

To show the use of PHP for CGI programs, we created a file in the `cgi-bin` directory of our V8.5.5 server that is named `helloWorld.php`. The code for our sample php CGI program is shown in Figure 12-11.

```
<?php
print <<<eot
<html>
<head>
<title>IBM ITS0 Simple php CGI</title>
</head>
<body>
<h1>A Simple php CGI from the ITS0</h1>
<p>Hello from a php CGI program</p>
</body>
</html>
eot;
?>
```

Figure 12-11 Our sample Perl CGI program

12.6.2 PHP wrapper program

To run PHP programs requires a *wrapper* program. We set up the wrapper program in the `cgi-bin` directory as shown in Figure 12-12. We saved this wrapper program in `/ihsconfig/ihs/ihsae002/cgi-bin/php`.

```
#!/bin/sh
/usr/lpp/php/5.1.2/bin/php_cgi "$@"
```

Figure 12-12 PHP wrapper program

12.6.3 Modifications to the httpd.conf file

The directives that were added to the httpd.conf file are shown in Figure 12-13. The AddHandler directive defines a name to be associated with any request that ends with .php. We used a name of php-script, but you can change this name to your own value.

```
# PHP support added here
# "php-script" is a name we get to make up
AddHandler php-script .php
# /cgi-bin/php is a URL, not a filesystem path
Action php-script /cgi-bin/php
# Debug output if the PHP stdout is bad
ScriptLog /ihsconfig/ihs/ihsae002/logs/php_stderr.log
```

Figure 12-13 Directives added to httpd.conf to support php

The Action directive tells V8.5.5 to start the wrapper program that is described in 12.6.2, “PHP wrapper program” on page 185 for requests that matched the AddHandler directive.

12.6.4 Testing the PHP CGI program

We used the following URL to start our PHP CGI program:

<http://wtsc55.itso.ibm.com:8235/cgi-bin/helloWorld.php>

The output that is shown in Figure 12-14 was produced.

```
A Simple php CGI from the ITS0

Hello from a php CGI program
```

Figure 12-14 Output from running the php CGI program

12.7 Lua support

This section describes the support for Lua that was added in IBM HTTP Server powered by Apache V9.0.

12.7.1 Lua overview

Lua is a powerful, fast, lightweight, embeddable scripting programming language that was developed in approximately 1993.

12.7.2 Lua and Apache server

Traditional CGI programming involves the use of some language, such as C, Perl, Rexx, or PHP. Regardless of what language you use for CGI programming, there is a key deficiency. When the Apache server receives a request to run a CGI program, it must spawn a new thread on which to run that CGI program.

On z/OS, this issue results in creating an OMVS Address Space to run the CGI program. Then, that address space is ended and cleaned up by z/OS at the completion of the CGI program.

For an Apache server that is running a light CGI load, this result might be OK. However, if your site is performing or planning to perform heavy CGI type processing, the current CGI approach is not ideal in terms of efficiency.

12.7.3 Lua advantage

The key advantage of Lua is that when the Apache server starts a Lua program, it runs on the thread in the Apache server that is processing the request. Therefore, no new thread is spawned and the processing is more efficient.

12.7.4 Using Lua

With any new technology that might be unfamiliar, one of the first questions asked by users often is: Is anyone of substance using this technology?

Lua is popular in the gaming industry, including the following popular titles:

- ▶ World of Warcraft
- ▶ Angry Birds

Lua also is used in Wireshark, which is a network protocol analyzer.

12.7.5 Lua examples

To enable Lua in our Apache server, we made several changes.

We changed the following line:

```
#LoadModule lua_module modules/mod_lua.so
```

to

```
LoadModule lua_module modules/mod_lua.so
```

We then found the following line in the httpd.conf file:

```
#AddHandler cgi-script .cgi
```

We added the following line after the line:

```
AddHandler lua-script .lua
```

We then created a file in the cgi-bin subdirectory and named it serverInfo.lua with the content that is shown in Example 12-17 on page 188.

Example 12-17 Sample Lua script

```
function handle(r)
  -- Set MIME type to text/plain:
  r.content_type = "text/plain"

  -- Call Function to get Host Name
  hostName = getHostName()
  r:puts("Hello from host: " .. hostName)
end
--
-- Function to get Host Name
--
function getHostName()
  return os.getenv("HOSTNAME")
end
```

We entered the following URL in a browser:

<http://wtsc55.itso.ibm.com:8229/cgi-bin/serverInfo.lua>

We saw the output that is shown in Example 12-18.

Example 12-18 Output from running sample Lua script

Hello from host: SC55

12.7.6 More information

For more information about Lua, see the following websites:

- ▶ <http://www.lua.org>
- ▶ <http://www.modlua.org>



A

Additional material

This paper refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material that is associated with this paper is available in softcopy on the Internet from the IBM Redbooks web server. Point your web browser at:

<ftp://www.redbooks.ibm.com/redbooks/REDP4987>

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select **Additional materials** and open the directory that corresponds with the IBM Redpaper publication form number: REDP4987.

Using the web material

The additional web material that accompanies this paper includes the following file:

<i>File name</i>	<i>Description</i>
redp4987.pdf	PDF of PowerPoint Presentation

System requirements for downloading the web material

The web material requires the following system configuration:

Hard disk space:	1 MB minimum
Operating System:	Windows
Processor:	Any
Memory:	1 MB

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material .zip file into this folder.



REDP-4987-02

ISBN 0738455598

Printed in U.S.A.

Get connected

