

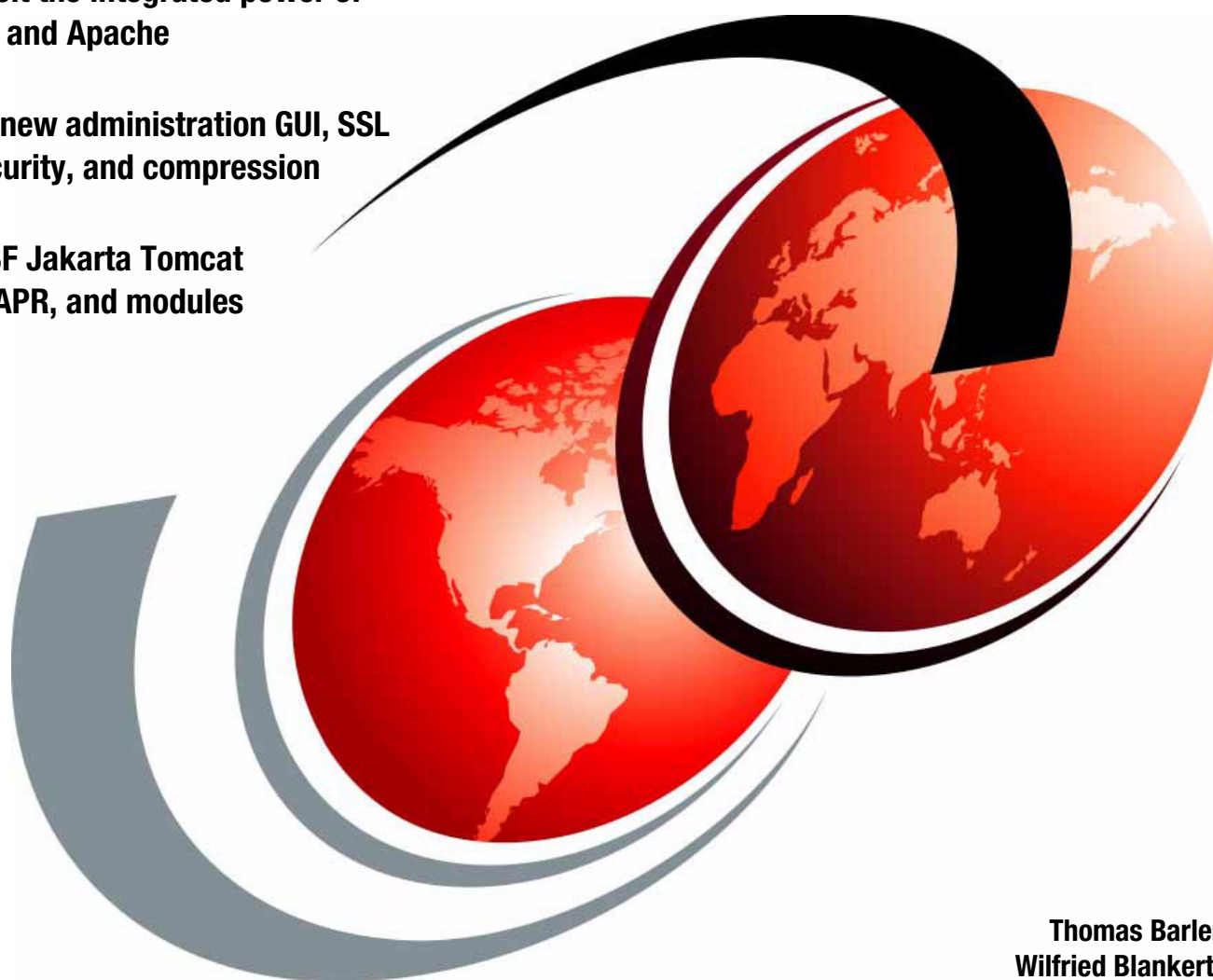
IBM HTTP Server (powered by Apache)

An Integrated Solution for IBM *e*server iSeries Servers

Fully exploit the integrated power of
IBM i5/OS and Apache

Study the new administration GUI, SSL
proxy, security, and compression

Extend ASF Jakarta Tomcat
5.5, PHP, APR, and modules



Thomas Barlen
Wilfried Blankertz

Redbooks



International Technical Support Organization

**IBM HTTP Server (powered by Apache): An
Integrated Solution for IBM @server iSeries Servers**

January 2005

Take Note! Before using this information and the product it supports, be sure to read the general information in “Notices” on page ix.

Third Edition (January 2005)

This edition applies to V5R3 of the IBM HTTP Server for iSeries (5722-DG1) for use with V5R3 of IBM i5/OS (5722-SS1).

© Copyright International Business Machines Corporation 2002, 2003, 2005. All rights reserved.

Note to U.S Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Contents	iii
Notices	ix
Trademarks	x
Foreword	xi
Preface	xiii
The team that wrote this redbook.	xiv
Become a published author	xv
Comments welcome.	xv
Summary of changes	xvii
January 2005, Third Edition	xvii
Part 1. Zen and the art of the HTTP server	1
Chapter 1. ‘Powered by Apache’ means OS/400 integration.	3
1.1 HTTP Server (powered by Apache) features	4
1.1.1 HTTP Version 1.1	5
1.1.2 GUI configuration and administration	5
1.1.3 Virtual hosts	6
1.1.4 Authentication	6
1.1.5 SSL and TLS	7
1.1.6 Proxy caching	7
1.1.7 Local memory cache	8
1.1.8 Server-side includes	9
1.1.9 CGI programming	9
1.1.10 LDAP support	9
1.1.11 Webserver Search Engine and Web Crawler	10
1.1.12 Web-based Distributed Authoring and Versioning	10
1.1.13 Access log reporting and Web usage mining	10
1.1.14 Log rollover and maintenance	11
1.1.15 Domino plug-in	11
1.1.16 WebSphere Application Server plug-in	11
1.1.17 Apache Software Foundation’s Jakarta Tomcat	11
1.1.18 Apache Portable Runtime and modules	12
1.1.19 Support for the TRCTCPAPP command	12
1.1.20 Collection Services performance data	12
1.1.21 Real-time server statistics	13
1.1.22 Triggered Cache Manager	13
1.1.23 Fast Response Cache Accelerator	13
1.1.24 Compression	13
1.1.25 Highly available HTTP server	14
1.1.26 Support for IASPs	14
1.1.27 Asynchronous I/O	14
1.1.28 Denial of service	14
1.1.29 Miscellaneous	15
1.2 For more information	15

Chapter 2. From zero to powered by Apache	17
2.1 Before you start	18
2.1.1 Software	18
2.1.2 User profile authorities	22
2.1.3 Web browser	22
2.2 Software installation	23
2.2.1 Installing LPPs and OS/400 options	23
2.2.2 Installing PTFs	24
2.2.3 Installing the ITSO example Web application (optional)	24
2.3 Testing the HTTP Server (powered by Apache) installation	24
2.3.1 Your first HTTP Server (powered by Apache) via a wizard	24
Chapter 3. The new GUI: IBM Web Administration for iSeries	33
3.1 Welcome page: iSeries Tasks page	34
3.2 Header images to access information for help	36
3.3 Tabbed pages for easy navigation	36
3.3.1 Setup tab: Common tasks and wizards	37
3.3.2 Manage tab	37
3.3.3 Advanced tab	51
3.3.4 Related links page	57
Chapter 4. Quick guide to Apache contexts and request routing	59
4.1 In-context configuration	60
4.2 Apache server request routing	61
4.3 Request routing example	62
4.4 Configuration recommendations	63
4.5 Configuring directory listings	63
Part 2. How to...	69
Chapter 5. Virtual hosts	71
5.1 HTTP virtual host overview	72
5.1.1 The way TCP/IP is configured	72
5.1.2 The way the HTTP server will be configured	72
5.1.3 The way the HTTP server will handle visitor requests	74
5.2 HTTP Server (powered by Apache) virtual host overview	75
5.2.1 Additional resources	77
5.3 Virtual hosts: IP-based implementation	77
5.3.1 IP-based virtual host: Problem scenario	78
5.3.2 IP-based virtual host: Solution overview	79
5.3.3 IP-based virtual host: Step-by-step implementation	80
5.4 Virtual hosts: Name-based implementation	89
5.4.1 Name-based virtual hosts: Problem overview	90
5.4.2 Name-based virtual host: Solution overview	91
5.4.3 Name virtual host: Step-by-step implementation	92
5.5 Virtual hosts: Mass dynamic implementation	94
5.5.1 Mass dynamic virtual host: Problem scenario	95
5.5.2 Mass dynamic virtual host: Solution overview	96
5.5.3 Mass dynamic virtual host: Step-by-step implementation	98
Chapter 6. Defending the IFS	101
6.1 Access control	102
6.2 Basic authentication	103
6.2.1 Authentication by OS/400 user profiles	105

6.2.2 Authentication by a validation list	108
6.2.3 Authentication by LDAP entries	113
6.3 Authenticating users via Kerberos	120
6.3.1 Getting ready for Kerberos authentication	122
6.3.2 Implementing Kerberos Web authentication	122
6.4 Encrypting your data with SSL and TLS	127
6.4.1 Enabling SSL	127
6.4.2 TLS upgrade	136
6.4.3 Enabling SSL for the ADMIN instance	137
6.4.4 SSL handshaking	137
6.4.5 Client-side digital certificates.	139
6.5 Proxy server: Protecting direct access	142
6.5.1 Forward proxy	143
6.5.2 Reverse proxy	145
6.5.3 SSL proxy	149
6.5.4 Proxy chaining	154
6.6 For more information.	155
Chapter 7. Serving dynamic data.	157
7.1 Server-side includes	158
7.2 Everything dynamic with CGI support.	160
7.3 Net.Data: A ready-made scripting tool	161
7.3.1 Implementation: Setting up the Net.Data environment	161
7.3.2 Configuring your HTTP Server (powered by Apache) for CGI	164
7.3.3 Testing your HTTP Server (powered by Apache) and Net.Data macro	169
7.4 For more information.	169
Part 3. Building a Web application	171
Chapter 8. Migration from HTTP Server (original) to (powered by Apache)	173
8.1 A look at HTTP Server (original) and (powered by Apache)	174
8.1.1 Directives and services not supported	175
8.1.2 Equivalent directives	176
8.1.3 Functional differences.	176
8.1.4 New HTTP Server (powered by Apache) directives	176
8.2 An example migration	177
8.2.1 Initial situation: HTTP Server (original) configuration	178
8.2.2 Migration steps	178
8.2.3 Result: HTTP Server (powered by Apache) configuration	186
8.3 Testing your migration.	188
Chapter 9. Web application serving	191
9.1 Web application servers for the iSeries server	193
9.1.1 Comparing WebSphere Application Server and ASF Jakarta Tomcat	194
9.1.2 When to use WebSphere Application Server versus ASF Jakarta Tomcat	195
9.2 Apache Software Foundation's Jakarta Tomcat on iSeries	197
9.2.1 ASF Jakarta Tomcat directory structure	198
9.2.2 ASF Jakarta Tomcat directives	199
9.2.3 ASF Jakarta Tomcat authorities	201
9.2.4 ASF Jakarta Tomcat log files	202
9.3 In-process implementation with ASF Jakarta Tomcat	202
9.3.1 Creating HTTP Server (powered by Apache)	202
9.3.2 In-process Tomcat configuration.	203
9.4 Out-of-process implementation with ASF Jakarta Tomcat	208

9.4.1	Creating the ASF Tomcat server.	209
9.4.2	Creating the link between the HTTP and ASF Tomcat servers	216
9.4.3	Testing the out-of-process ASF Tomcat server	220
Chapter 10.	Getting the best performance from HTTP Server (powered by Apache)	223
10.1	iSeries Web server performance components	226
10.2	Web server: Global performance values.	227
10.2.1	Threads and asynchronous I/O.	228
10.2.2	Process control: HotBackup	229
10.2.3	Logging	230
10.2.4	HostNameLookups	231
10.2.5	KeepAliveTimeout.	231
10.2.6	TCP buffer size	232
10.2.7	Denial of service	233
10.2.8	CGI initialization at server startup.	234
10.3	Web server: Specific performance values.	235
10.3.1	HTTP Server (powered by Apache) local cache.	236
10.3.2	HTTP Server (powered by Apache) proxy cache	239
10.4	Increasing throughput with compression.	240
10.4.1	Compression considerations.	241
10.4.2	Example configurations.	241
10.4.3	Logging	252
10.4.4	Controlling the compression environment.	257
10.4.5	For more information.	258
10.5	Triggered Cache Manager	259
10.5.1	TCM system requirements	260
10.5.2	TCM documentation	261
10.5.3	TCM directory structure and authorization	261
10.5.4	How the TCM server works.	262
10.5.5	Configuring a working TCM example	264
10.6	Fast Response Cache Accelerator	281
10.6.1	What FRCA is	282
10.6.2	How FRCA local cache works.	283
10.6.3	How FRCA reverse proxy cache works	285
10.6.4	FRCA limitations	286
10.6.5	FRCA configuration examples	287
10.6.6	Miscellaneous FRCA directives beyond the online help	296
10.6.7	The FRCA challenge.	299
10.6.8	For more information.	299
10.7	Cryptographic coprocessors	300
10.8	Real Time Server Statistics.	301
10.9	References	306
Chapter 11.	Getting started with Webserver Search Engine and Web Crawler.	307
11.1	iSeries Webserver Search Engine	308
11.2	iSeries Webserver Search Engine Web Crawler.	309
Chapter 12.	Apache Portable Runtime: Extending your core functionality.	311
12.1	Apache module design overview	312
12.1.1	Documentation and resources	314
12.2	Creating a module for the iSeries server.	315
12.2.1	The task at hand	315
12.2.2	Source code and comments	315
12.2.3	Compiling, linking, and exporting your service program	319

12.2.4	Activating via configuration	320
12.2.5	Testing header_module	320
12.2.6	Debugging	322
Chapter 13.	Problem determination: When things do not go as planned	323
13.1	The art of problem determination	324
13.2	Tools of the trade	327
13.2.1	Working with configuration files	327
13.2.2	Job logs	329
13.2.3	Server logs	331
13.2.4	Net.Data logs and traces	340
13.2.5	HTTP server trace	341
13.2.6	Collection Services performance data	345
13.2.7	Other startup parameters	351
13.2.8	HTTP status codes	352
13.2.9	Communications trace	353
13.2.10	Additional resources	354
Chapter 14.	High availability	355
14.1	Highly available Web server cluster on the HTTP server	356
14.1.1	Primary or backup with takeover IP model	356
14.1.2	Primary or backup with a network dispatcher model	358
14.1.3	Peer model	359
14.2	A working primary or backup with takeover IP model	359
14.2.1	Problem definition	359
14.2.2	Solution definition	360
14.2.3	Assumptions	360
14.2.4	How to	361
14.3	For more information	371
Chapter 15.	National language considerations	373
15.1	Installing secondary languages	374
15.2	Net.Data based: iSeries Tasks page and DCM	375
15.3	Servlet based: Administration GUI	376
15.4	Other programs linked from iSeries Task page	380
15.4.1	Internet Printing Protocol server for the iSeries server	380
15.4.2	WebSphere family	381
15.4.3	4758 Cryptographic Coprocessor	381
15.5	Serving your own Web site in the world's languages	381
Part 4.	Appendixes	385
Appendix A.	Bringing PHP to your iSeries server	387
Programming with PHP on the iSeries server		388
What PHP is		388
Why PHP		389
A code example		390
PHP on the iSeries server		391
PHP as a CGI program		393
Another PHP script		396
For more information		397
Beware of PHP bugs		397
Prerequisites		398
Installing PHP on the iSeries server		399

Pre-preparation for installation	399
Downloading PHP	400
Patching the source code file	401
Locating iSeries-specific files	401
Preparing for the PHP compile	402
Compile (make).	403
Testing PHP	405
Configuring HTTP Server (powered by Apache) to use PHP	405
Creating a sample database	405
Limitations	406
PHP 4.2.2 errata	407
Appendix B. Bringing Tomcat Version 5.5 to your iSeries server	409
Software prerequisites	410
Installation	410
Installing Tomcat 5.5 on your iSeries server	411
Installing the Tomcat 5.5 compatibility package	412
Starting Tomcat 5.5 on the iSeries server	413
Installing mod_jk connector	415
Configuring your HTTP Server (powered by Apache).	416
Appendix C. Bringing Zip and Unzip to OS/400 PASE and Qshell environments . . .	419
Appendix D. Additional material	421
Locating the Web material	421
Using the Web material	421
System requirements for downloading the Web material	422
How to use the Web material	422
Related publications	423
IBM Redbooks	423
Other resources	424
Referenced Web sites	424
How to get IBM Redbooks	426
Help from IBM	426
Index	427

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law. INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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.

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 the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates 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. 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 IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®
alphaWorks®
developerWorks®
eServer™
ibm.com®
iSeries™
i5/OS™
pSeries®
AIX®
AS/400®

Domino®
DB2 Universal Database™
DB2®
Integrated Language Environment®
IBM®
Language Environment®
Lotus®
Net.Data®
OS/400®
PartnerWorld®

PowerPC®
POWER™
Redbooks™
Redbooks(logo) ™
RS/6000®
SecureWay®
Tivoli®
TotalStorage®
VisualAge®
WebSphere®

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, 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 Sun Microsystems, Inc. in the United States, other countries, or both.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Foreword

IBM® embraced the widely popular open-source Apache server several years ago as the Hypertext Transfer Protocol (HTTP) server of choice for its Web products. The foundation of any On Demand Business application is the Web server, and IBM has made a significant investment in Apache to be that foundation. The broad investment in Apache across IBM's product offerings allows Web developers to leverage existing Apache skills and software to build applications for commercial use.

Just as your businesses do not stand still, neither does the world of Web products. The Apache 2.0 server continues to evolve to meet the demands of today's Web environments. The IBM HTTP Server for iSeries™ (powered by Apache) continues to evolve as well, incorporating the latest Apache changes into the IBM @server i5 and iSeries platform and integrating them in an easy-to-use and easy-to-manage fashion. By reading this IBM Redbook, you will gain a sense for the breadth of the functions available in the HTTP Server (powered by Apache), as well as learn how to integrate the HTTP Server with the rest of your computing systems and integrate your Web environment with IBM i5/OS™ applications. You will understand how the renowned i5/OS security has been leveraged within the HTTP Server (powered by Apache). You will also see how convenient all these capabilities are to configure and use thanks to the administrative graphical user interface (GUI) that is provided.

If you are familiar with this redbook from earlier versions, you will find that the features of the HTTP Server (powered by Apache) continue to grow. More options and greater integration with Web solutions are provided with the addition of Kerberos authentication. New performance enhancements are available with data compression.

If you are new to this redbook, you will find a wealth of information about what is possible with the HTTP Server (powered by Apache) and to leverage these rich features for your benefit. Examples and details are provided so that you can duplicate what we have done and take it from there.

Evolving right along with the server is the administrative GUI. The administrative GUI is another example of integration within the iSeries system to enable you to achieve the most from your Web environment.

Whether you are looking to enhance an existing Web environment or introduce a new Web solution, we hope that this redbook provides the information, or references to information, to help you capitalize on the functionality provided with IBM HTTP Server for iSeries (powered by Apache).

Brian Noordyke

HTTP Server Development Team Leader
IBM @server i5

Preface

This IBM Redbook is designed as a guide to help you plan, install, configure, troubleshoot, and understand the IBM HTTP Server (powered by Apache) running on the IBM *eServer* i5 and iSeries server. This redbook starts with an introduction to the HTTP Server (powered by Apache). It identifies all the components that are necessary for you to install and configure your first Apache-based Web server running on your iSeries server. It includes a quick guide to the Apache contexts and request routing. It also introduces the iSeries' unique graphical user interface (GUI) for further configuration and customization.

Then this redbook instructs you on how to use virtual hosts, secure your server, and serve dynamic data with server-side includes (SSI), Common Gateway Interface (CGI), Net.Data®, and Hypertext Preprocessor (PHP). Each lesson is written in an easy to follow "how to" style.

After that, this redbook takes an in-depth look at the HTTP Server (powered by Apache). It details the steps that are necessary to implement Web application serving with Java™, featuring the Apache Software Foundation's (ASF) Jakarta Tomcat. More advanced topics include how to achieve the best performance by using local caches, compression with mod_deflate, Triggered Cache Manager (TCM), and the Fast Response Cache Accelerator (FRCA).

One of the key differentiators of i5/OS compared to most other operating systems is the many built-in security features and services that make this platform one of the most secure platforms in the market. The HTTP Server (powered by Apache) in i5/OS also includes many security functions extending the i5/OS security to the Web environment. An entire part of this redbook is devoted to protecting your Web server and data traffic.

This redbook also introduces the Webserver Search Engine, problem determination, high availability (HA), and national language support (NLS) considerations. It includes an example of extending the core features of your HTTP Server (powered by Apache) via Apache Portable Runtime (APR) support. This allows you to write your own modules or port them to the iSeries as Integrated Language Environment® (ILE) service programs.

To complete the discussion, this IBM Redbook concludes with appendixes about bringing PHP and Tomcat Version 5.5 to your i5/OS operating system. It also includes an appendix about bringing zip and unzip functions to the OS/400® Portable Application Solutions Environment (OS/400 PASE) and Qshell environments.

As an added bonus, you can download all the examples provided in this redbook from the Web as explained in Appendix D, "Additional material" on page 421. This allows you to reduce the transition time from understanding to implementation.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the IBM International Technical Support Organization (ITSO), Rochester Center.



Thomas Barlen is an IBM Certified Consulting IT Specialist for iSeries systems in IBM @server iSeries Technical Sales Germany. His current areas of expertise include i5/OS security, single signon, networking, Linux integration on iSeries, and On Demand Business infrastructure. Before joining this team in September 2002, Thomas was assigned to the ITSO, Rochester Center. He writes extensively and teaches IBM classes worldwide on all areas of iSeries communications, On Demand Business infrastructure, and security. Thomas is also a frequent speaker at technical conferences around the globe. Prior to his start in the ITSO in 1999, he worked in AS/400® software support and as a systems engineer in IBM Germany. He has over 15 years of experience in AS/400 and iSeries networking and system management, as well as LAN and WAN network design and implementation. You can reach Thomas by sending e-mail to barlen@de.ibm.com.



Wilfried Blankertz is a Senior IT Specialist for iSeries Technical Sales in the IBM EMEA Central region located in Frankfurt, Germany. From 1995 to 1998, he was assigned to the ITSO, Rochester Center, where he coauthored more than 14 IBM Redbooks™ and taught IBM classes worldwide on all areas of OS/400® Groupware solutions and Systems Management. Before joining the ITSO, he worked as a systems engineer in IBM Germany supporting customers with the AS/400 system and its predecessor systems for 30 years. He is also a Certified Lotus® Professional for Administration and for Domino® R5 Application Development. He also owns several certifications for WebSphere® application development for iSeries. You can reach Wilfried by sending e-mail to WilBlank@de.ibm.com

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

Brian R. Smith, IBM Rochester, for his excellent ideas for, and efforts in, creating the first two editions of this redbook

Wade Fode
Terry Hennessy
Brian Krings
Brian Noordyke
Scott McCreadie
Ryan Pendergast
Karen L. Richner
IBM Rochester

Debbie Landon
ITSO, Rochester Center

We extend a special thank you to the following contributors:

- ▶ Henri Gomez of SLIB, France, Apache Tomcat Project Committer, who provides us with the Jakarta Tomcat Connectors for the iSeries. For more information, see:
<http://jakarta.apache.org/builds/jakarta-tomcat-connectors/jk/release/>
- ▶ The previous authors of this redbook:
Gaia Banchelli
Monica Maria Echeverry
Axel Lachmann
John Nesbitt
Wolfgang Pauer
Brian R. Smith

Sections of this IBM Redbook were prepared with assistance from Information Development at IBM Rochester.

With permission from iSeries Network, we include in this IBM Redbook an article that was previously published online at:

<http://www.iseriesnetwork.com>

See “Programming with PHP on the iSeries server” on page 388. In addition, “Fast Response Cache Accelerator” on page 281 is also largely based upon a two-article series written for iSeries Network.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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 Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

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

ibm.com/redbooks

- Send your comments in an Internet note to:

redbook@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. JLU Building 107-2
3605 Highway 52N
Rochester, Minnesota 55901-7829

Summary of changes

This section describes the technical changes made in this edition of the book. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes for *IBM HTTP Server (powered by Apache): An Integrated Solution for IBM @server iSeries Servers*, SG24-6716-02, as created or updated on January 6, 2005.

January 2005, Third Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

- ▶ An overview of new features has been added to Chapter 1, “Powered by Apache’ means OS/400 integration” on page 3.
- ▶ Basic authentication with users in Lightweight Directory Access Protocol (LDAP) directories is a common authentication method for Internet users. IBM i5/OS V5R3 provides an entirely new LDAP management interface. Chapter 6, “Defending the IFS” on page 101, has been changed to include the instructions to set up LDAP-based basic authentication for V5R3 including the Web-based LDAP management utility to manage users.
- ▶ A new authentication method that has been added to the HTTP Server (powered by Apache) is the Kerberos network authentication mechanism. Kerberos authentication provides single signon capabilities and automatic authentication for protected Web resources. Chapter 6, “Defending the IFS” on page 101, includes the setup information to configure the HTTP Server (powered by Apache) to authenticate users via Kerberos tickets.
- ▶ Confidentiality is a major goal in security. Using encryption functions, such as Secure Sockets Layer (SSL), you can ensure that network traffic cannot be read by an unauthorized user while in transit. The HTTP Server (powered by Apache) now provides SSL proxy support for a reverse proxy environment. Detailed instructions have been added to Chapter 6, “Defending the IFS” on page 101, that explain how to set up the reverse SSL proxy.
- ▶ A good performing HTTP server is key to run a successful On Demand Business. Real Time Server Statistics are a new function that provide information about the current behavior and utilization of the HTTP Server (powered by Apache). Information about the new support have been added to Chapter 1, “Powered by Apache’ means OS/400 integration” on page 3, and Chapter 10, “Getting the best performance from HTTP Server (powered by Apache)” on page 223.

Changed information

- ▶ Feature changes, release information, and PTF level information have been updated in all chapters.
- ▶ Most window captures have been updated in the entire book to reflect the new graphical user interface (GUI) as introduced in 2004.

- ▶ One method to serve dynamic Web content to browsers is to use Common Gateway Interface (CGI) programs. Information has been added to Chapter 7, “Serving dynamic data” on page 157, about the support of Java CGI programs.
- ▶ Information about the withdrawn HTTP Server (original) has been removed from several chapters in the book.
- ▶ Chapter 13, “Problem determination: When things do not go as planned” on page 323, has been updated to include more debugging help. This includes more information about error messages and possible causes, as well as how to solve problems with the HTTP Server (powered by Apache).
- ▶ PHP is one of the major scripting languages. Appendix A, “Bringing PHP to your iSeries server” on page 387, includes updated information about how to obtain binary PHP packages for OS/400 and i5/OS.
- ▶ ASF Jakarta Tomcat is an open source product that allows you to run JavaServer Pages (JSPs) and Servlets. Appendix B, “Bringing Tomcat Version 5.5 to your iSeries server” on page 409, has been changed to provide instructions for installing and setting up Tomcat Version 5.5 for the HTTP Server (powered by Apache).



Part 1

Zen and the art of the HTTP server

Your iSeries Web server is the center of most all On Demand Business applications. Hypertext Transfer Protocol (HTTP) is the protocol used to communicate between a client (browser) and your Web server. HTTP can be used to carry the order from your customers and allow you to respond with a “thank you”.

Network administrators know how to configure your firewalls to allow the HTTP protocol between your private intranet and the public Internet. They also know how to force people to sign on before they access sensitive portions of your Web site. And they know how to encrypt the data using powerful protocols such as Secure Sockets Layer (SSL) and Transport Layer Security (TLS). These powerful tools are available to your network administrator via a graphical user interface (GUI) configuration, not through complex programming.

Application servers, such as IBM WebSphere Application Server and the Apache Software Foundation’s Jakarta Tomcat, can allow you to dynamically extend the power of your HTTP server core features and functions. They are the incarnation of specifications from Sun Microsystems that allow you to program industry standard Java applications as servlets and JavaServer Pages (JSPs).

Read on to learn more about how the HTTP Server (powered by Apache) can handle your On Demand Business needs. As you will see, this story is all about the integration of the Apache Web server with OS/400.



‘Powered by Apache’ means OS/400 integration

Most Hypertext Transfer Protocol (HTTP) servers originate from CERN or National Center for Supercomputing Application (NCSA). The Apache server originates from NCSA. The fundamental ideas behind and the basic design of the World Wide Web evolved from work being done at CERN in Geneva, Switzerland. In its roots, the Apache server was developed at NCSA, and it was based on the NCSA HTTP daemon (NCSA HTTPd 1.3).

The NCSA Web server, at that time, was adopted and used by a large number of webmasters in the market. In mid-1994, however, the development for this Web server stalled and left many webmasters to find their own solutions to problems encountered in their environments. Some of them developed their own extensions and problem fixes, which could apply to other webmasters searching for the same solution.

In February 1995, a group of webmasters volunteered to consolidate all information related to the server and placed it in a publicly accessible domain for all webmasters to access. The Apache Group was then formed from people who made substantial contributions to the Apache server. NCSA later revived the suspended development of their NCSA Web server, and two members from that development team joined the Apache Group so that ideas and contributions could be exchanged among both projects. The Apache Group reviewed some of the enhancements and bug fixes and added them to their own server for testing purposes.

In April 1995, the Apache server made its first public release with Version 0.6.2. It was given this name because it was the “patched” version (A PAtCHy server) of the NCSA HTTPd 1.3 Web server.

From May through June 1995, some general overhaul and redesign was made to fine-tune the Apache server, along with the introduction of some new features in Version 0.7.x. The next release of the Apache server with Version 0.8.8 in August 1995 brought about a change in the architecture of the server with the modular structure and application programming interface (API) features. The latest level available for the Apache server is Version 2.0. It is this version that we enjoy on the iSeries with the HTTP Server (powered by Apache).

The market share for the top servers across all domains in October 2004 was:

- ▶ 67.92% for Apache
- ▶ 21.09% for Microsoft® IIS
- ▶ 3.04% for SunONE
- ▶ 1.35% for Zeus

Source: For more information, see the Netcraft survey from September 2004 at:

<http://news.netcraft.com/>

Apache, a freeware HTTP server, is open-source software that implements the industry standard HTTP/1.1 protocol. The focus is on being highly configurable and easily extendable. It is built and distributed under the Apache Software Foundation (ASF). It is available on the Web at:

<http://www.apache.org>

The benefit of Apache to iSeries users is that the HTTP Server (powered by Apache) is based on the open-source server code provided by the Apache Software Foundation. This version is based on the general availability (GA) code for Apache Version 2.0. It is updated as future Apache versions are made available. While iSeries source code not published, IBM offers any enhancements it develops to the Apache Software Foundation in an open-source form for inclusion in the Apache server. As with any supported product, IBM provides defect support for the HTTP Server (powered by Apache). IBM has long been active in Apache development.

Tip: At the time of writing this book, the HTTP Server (powered by Apache) code on the iSeries was based upon Apache Version 2.0.49. To see which version of Apache code is the base for your HTTP Server (powered by Apache), see 13.2.7, “Other startup parameters” on page 351.

This integration with OS/400 forces IBM to no longer call this server an Apache server. It must be called the *HTTP Server (powered by Apache)* with the parenthetical phrase as a bold reminder of the power and value of the integration with OS/400.

1.1 HTTP Server (powered by Apache) features

Previously, OS/400 provided two different HTTP servers. The first one was the HTTP Server (original) and was available up to OS/400 V5R2. Starting with IBM i5/OS V5R3, the only HTTP server that is available for the operating system is the HTTP Server (powered by Apache). Both servers have similar functions, but all newer functions were introduced only to the HTTP Server (powered by Apache). This section provides a functional overview of the HTTP Server (powered by Apache).

The overview is based on the V5R3 version of the HTTP Server (powered by Apache). Using program temporary fixes (PTFs), the Rochester lab sent back much of this HTTP Server (powered by Apache) functionality to V5R1 and V5R2. In essence, the overview is true for V5R1 with the exception of these features that were not sent back via PTF:

- ▶ Fast Response Cache Accelerator (FRCA)
- ▶ Collection Services performance data
- ▶ Support for independent auxiliary storage pools (IASPs)
- ▶ Real-time server statistics
- ▶ Kerberos user authentication

Tip: OS/400 V5R2 was the last release that supported the HTTP Server (original). Enhanced migration options are available with the HTTP Server migration wizard of the IBM Web Administration for iSeries interface that can help you migrate your existing configuration to a HTTP Server (powered by Apache) configuration. For more information about migrating HTTP Server (original) configurations to HTTP Server (powered by Apache), refer to the migration article:

<http://www.ibm.com/servers/eserver/iseries/software/http/product/migrate.html>

The HTTP Server (powered by Apache) is the better choice. In this redbook, see Chapter 8, “Migration from HTTP Server (original) to (powered by Apache)” on page 173.

1.1.1 HTTP Version 1.1

The HTTP Server (powered by Apache) supports HTTP Version 1.1 (written as HTTP/1.1). The HTTP protocol implementation in Apache was chiefly architected by one of the HTTP/1.1 authors. Most current versions of popular Web browsers support HTTP/1.1. Apache is normally configured to detect popular browsers that do not properly support HTTP/1.1 and use only HTTP/1.0.

Persistent connections

When you enter a Uniform Resource Locator (URL) into your browser's address line or click a link on a Web page, you open a connection between your browser and the HTTP server. Prior to the availability of persistent connections, each file referenced on the Web page was retrieved using a separate connection. This type of retrieval is tremendously costly for the HTTP server and the network since overhead is required to establish and terminate each connection. Persistent connections are the default behavior for an HTTP server that implements the HTTP/1.1 protocol. They allow retrieval if there are multiple elements within a single connection.

1.1.2 GUI configuration and administration

You can configure and administrate HTTP server instances from Web browsers. To access the iSeries Tasks page, start the administration server:

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
```

You may also start the administration server using iSeries Navigator (formerly called Operations Navigator) as explained in 2.3.1, “Your first HTTP Server (powered by Apache) via a wizard” on page 24.

Then type the following URL from a Web browser:

```
http://hostname:2001
```

This brings you to the iSeries Tasks page. From there, click **IBM Web Administration for iSeries**. This option allows you to configure the HTTP Server (original) (up to V5R2) and the HTTP Server (powered by Apache), WebSphere Application Server Express or Base, Version 5 or later, and other Web-related functions.

Tip: The administration server's port 2010 can be configured for a Secure Sockets Layer (SSL) or Transport Layer Security (TLS) encrypted session. See 6.4.3, “Enabling SSL for the ADMIN instance” on page 137.

Up to OS/400 V5R2, the HTTP Server (original) and the HTTP Server (powered by Apache) can coexist. That is, you may have zero, one, or many HTTP Server (original) servers running at the same time that you have zero, one, or many HTTP Server (powered by Apache) servers running. The administration page allows you to create and manage HTTP servers. However, with i5/OS V5R3 and later, the HTTP Server (original) is no longer supported and must be migrated to the HTTP Server (powered by Apache). For more information, see Chapter 8, “Migration from HTTP Server (original) to (powered by Apache)” on page 173.

The HTTP Server (powered by Apache) administration graphical user interface (GUI) is available in multiple languages. For information about configuring your HTTP Server (powered by Apache) in your native language, see 15.3, “Servlet based: Administration GUI” on page 376.

1.1.3 Virtual hosts

You can enable virtual hosting. This allows you to host any number of Web sites through one communications adapter. With virtual hosting, you do not need to assign a unique port to each Web site. Virtual hosting is useful if you need to provide multiple “top-level” URLs for your Web sites or if you provide Internet Service Provider (ISP) services to clients.

See Chapter 5, “Virtual hosts” on page 71, for more information about the more flexible solution for the HTTP Server (powered by Apache).

Dynamic virtual hosting

The dynamic virtual host allows you to dynamically add Web sites (host names) by adding directories of content. This approach is based on automatically inserting the Internet Protocol (IP) address and the contents of the Host: header into the path name of the file that is used to satisfy the request.

See 5.5, “Virtual hosts: Mass dynamic implementation” on page 94, for more information.

1.1.4 Authentication

Several options are available to authenticate Web users when accessing protected resources.

Basic authentication

Basic authentication is a popular way to secure Web resources. You can protect Web resources by asking the user for a user ID and password to gain access to these resources. Specifically for the HTTP Server (powered by Apache), the user ID and password on the iSeries can be validated in one of three ways:

- ▶ OS/400 user profile: This requires each user to have a system user profile.
- ▶ Validation list: This requires you to create a validation list that contains Internet users.
- ▶ Lightweight Directory Access Protocol (LDAP) server: This requires you to configure an LDAP server with the user entries.

For more information, see 6.2, “Basic authentication” on page 103.

Kerberos authentication

Introduced via PTFs in V5R2 and included in V5R3, the HTTP Server (powered by Apache) also supports authentication via the Kerberos authentication protocol. This technology is used in single signon environments and uses tickets instead of user identifiers and passwords. For the HTTP Server (powered by Apache), Kerberos is used in conjunction with Enterprise Identity Mapping (EIM) to authenticate Kerberos tickets and map the Kerberos user principal to OS/400 user profiles. This authentication method is an excellent choice in an intranet environment. For more information, see 6.3, “Authenticating users via Kerberos” on page 120.

Client-side certificate authentication

The third option to authenticate individual users to protected resources is the authentication via digital certificates. This authentication mechanism requires the HTTP Server (powered by Apache) to be configured for SSL. You can find more information in 1.1.5, “SSL and TLS” on page 7, and 6.4.5, “Client-side digital certificates” on page 139.

1.1.5 SSL and TLS

SSL has become an industry standard for enabling applications for secure communication sessions over an unprotected network (such as the Internet). With the SSL protocol, you can establish secure connections between clients and server applications which provide authentication of one or both end points of the communication session. SSL also provides privacy and integrity of the data that client and server applications exchange. Multiple versions of the SSL protocol are defined. The latest version, Transport Layer Security Version 1.0, provides an evolutionary upgrade from SSL Version 3.0.

HTTP Server (powered by Apache) supports server authentication and client authentication using digital certificates. With *server authentication*, the client ensures that the server certificate is valid and that it is signed by a Certificate Authority (CA) which the client trusts. With *client authentication*, the server ensures that the client certificate is valid and that it is signed by a CA which the server trusts.

For more information, see 6.4, “Encrypting your data with SSL and TLS” on page 127.

SSL proxy

A recent enhancement to the HTTP Server (powered by Apache) is the support of an SSL proxy server. This function is typically used as a reverse proxy where clients from the Internet access resource behind a proxy on a server in an intranet or DMZ. Prior to this enhancement, reverse proxy connections would not support client-to-content server SSL connections. For more information, see 6.5.3, “SSL proxy” on page 149.

1.1.6 Proxy caching

You can configure IBM HTTP Server for iSeries as a non-caching or caching proxy server. When used as a non-caching proxy, the primary benefit of enabling proxy services is that the IP addresses used on the internal network are not sent out of your network. The proxy service forwards the request from your internal network using the IP address of the proxy server, not the address of the original requester. When the proxy server receives the response, it forwards the response to the original requester.

With caching enabled, the proxy server can act as a high-speed local store of previously accessed Web pages. When you configure the server as a proxy caching server, you can improve performance. You can also allow users of your internal network to access documents on the Internet. For example, if you frequently access the same set of Web pages from one or more sites, it may be advantageous to activate the caching feature. The retrieved Web page is stored locally on your iSeries server. Any subsequent accesses to the page occur at local area network (LAN) speed, rather than at Internet speed.

Web pages can be encoded with a “no-cache” attribute or a specific expiration date. You can also configure the IBM HTTP Server for iSeries proxy service so that it periodically performs “garbage collection” to remove expired files from the cache.

Another use of the proxy service (with or without caching) is to log client requests. Some of the available data includes:

- ▶ Client IP address
- ▶ Date and time
- ▶ URL requested
- ▶ Byte count
- ▶ Success code

With this information, you can construct reports to account for the use of your Web site. For example, the information can be used in a charge-back system to understand and track marketing trends.

See 6.5, “Proxy server: Protecting direct access” on page 142, for security-related information about proxy support, and 10.3.2, “HTTP Server (powered by Apache) proxy cache” on page 239, for performance-oriented information about caching.

Reverse proxy caching

Reverse proxy is another common form of a proxy server. It is generally used to pass requests from the Internet, through a firewall, to isolated, private networks. It is used to prevent Internet clients from having direct, unmonitored access to sensitive data residing on content servers on an isolated network, or intranet. If caching is enabled, a reverse proxy can also reduce network traffic by serving cached information, rather than passing all requests to actual content servers. Reverse proxy servers may also balance workload by spreading requests across a number of content servers.

An advantage of using a reverse proxy is that Internet clients do not know their requests are being sent to and handled by a reverse proxy server. This allows a reverse proxy to redirect or reject requests without making Internet clients aware of the actual content server (or servers) on a protected network.

For more information, see 6.5.2, “Reverse proxy” on page 145.

In addition the HTTP Server (powered by Apache) can use FRCA. FRCA incorporates both a local cache and another reverse proxy cache. That is, both the HTTP Server (powered by Apache) and FRCA have the ability to be configured as a reverse proxy cache. For information about FRCA, see 10.6, “Fast Response Cache Accelerator” on page 281.

1.1.7 Local memory cache

A proxy cache is traditionally most beneficial to clients on your network since it enables you to store files that were retrieved from other Web sites. You can provide a caching service for files on your site using the local memory cache configuration options.

To use a local memory cache, you identify an amount of memory to allocate and a set of files to be cached. When the IBM HTTP Server for iSeries is started, the files are read into the local memory cache up to the limit of the amount of memory allocated or the limit of the number of files that you allow to be cached.

When a request is received at your IBM HTTP Server for iSeries, the local memory cache is checked first to determine if it has a copy of the requested file. If so, the file is served from the cache, which can be significantly faster than if the file is retrieved from disk storage.

See 10.3.1, “HTTP Server (powered by Apache) local cache” on page 236, for more information. In addition, the HTTP Server (powered by Apache) (only) supports FRCA which also has a local cache option.

1.1.8 Server-side includes

Server-side includes (SSI) enable the server to process some of the Web pages before the server sends the page to the client. The current date, size of the file, and the last change date of a file are examples of the kind of information that you can include in Web pages that you send to the client.

See 7.1, “Server-side includes” on page 158, for more information.

1.1.9 CGI programming

Corporations and other customers benefit from interacting with browser users by sending and receiving data. In the Web presence arena, this type of transaction is simple, such as collecting the name and address of a browser user who wants to receive a catalog. In general, these interactions start with a form, a Web page that contains input-capable fields and push buttons (like function keys). The server needs to hand the input from the form to a program for processing.

Typically, on the iSeries server (and most other platforms), this program is a Common Gateway Interface (CGI) program. The CGI program receives the form data from the browser, accesses business data and business logic on the iSeries server, updates or stores information (if required by the transaction), and then builds the Web page that the HTTP server returns to the browser user in response.

CGI programs written for the HTTP Server (original) function the same way for the HTTP Server (powered by Apache).

In addition, CGI applications working with the HTTP Server (powered by Apache) can:

- ▶ Control the number of CGI jobs started with the server and their user profile
- ▶ Run Portable Application Solutions Environment (OS/400 PASE) (UNIX® binaries) applications as CGI programs

See 7.2, “Everything dynamic with CGI support” on page 160, for more information.

1.1.10 LDAP support

The HTTP servers can use an LDAP-enabled directory to store:

- ▶ **Configuration information:** Refer to *Implementation and Practical Use of LDAP on the IBM® iSeries Server*, SG24-6193, for information about how to use LDAP to store (and share) HTTP server configurations throughout your network.

- **User authentication information:** See 6.2.3, “Authentication by LDAP entries” on page 113, to learn about the security aspect of LDAP configuration.

1.1.11 Webserver Search Engine and Web Crawler

The HTTP Server search engine allows you to perform full text searches on Hypertext Markup Language (HTML) and text files stored in the iSeries file system from any Web browser. The iSeries Webserver Search Engine is available at no charge with IBM HTTP Server for iSeries (5722-DG1). You can control the options that are available to the user and how the search results are displayed through customizable Net.Data macros.

On the iSeries server, the search engine comes in two logical pieces that are related to each other. You can read more about them in 11.1, “iSeries Webserver Search Engine” on page 308, and 11.2, “iSeries Webserver Search Engine Web Crawler” on page 309.

1.1.12 Web-based Distributed Authoring and Versioning

Web-based Distributed Authoring and Versioning (WebDAV) provides a network protocol for creating interoperable, collaborative applications. Major features of the protocol include:

- Locking (concurrency control)

Long-duration exclusive and shared write locks prevent the problem of overwriting, where two or more collaborators write to the same resource without first merging changes. To achieve robust Internet-scale collaboration, where network connections may be disconnected arbitrarily, and for scalability, since each open connection consumes server resources, the duration of DAV locks is independent of any individual network connection.

- Properties

Extensible Markup Language (XML) properties provide storage for arbitrary metadata, such as a list of authors on Web resources. These properties can be efficiently set, deleted, and retrieved using the DAV protocol. The DAV Searching and Locating (DASL) protocol provides searches based on property values to locate Web resources.

- Namespace manipulation

Since resources may need to be copied or moved as a Web site evolves, DAV supports copy and move operations. Collections, similar to file system directories, may be created and listed.

For more information about WebDAV, see the following Web site:

<http://www.webdav.org/>

You may also want to refer to Request for Comments (RFC) 2518: HTTP Extensions for Distributed Authoring – WEBDAV, which you can find on the Web at:

<http://www.ietf.org/rfc/rfc2518.txt>

1.1.13 Access log reporting and Web usage mining

The HTTP Server (original) provides the log reporting and Web usage mining function. If you are using HTTP Server (powered by Apache), you can obtain the IBM WebSphere Site Analyzer to provide a similar function.

Tip: The market is full of Web log analyzer software that supports the industry-standard common and combined log formats for Apache. The HTTP Server (powered by Apache) uses these same formats. Choose the one that you feel is the best value for your money.

The HTTP Server (powered by Apache) includes complete facilities to log client access, server errors, and other forms of customizable information. See 13.2.3, “Server logs” on page 331, for more information.

1.1.14 Log rollover and maintenance

The HTTP Server (original) supports daily log files only. When a server instance is started, all of the log files configured for that server instance are opened. By default, the server does not create any logs. The proper directives must be configured by the Web administrator to cause the HTTP server to log. Web server instances may not share log files.

The Apache code, as shipped from ASF, has no automatic rollover capability. If the user wants the current log rolled, the support must be implemented via a user program.

In V5R2, the iSeries extended this feature with the HTTP Server (powered by Apache) to include log rollover support. This is in the form of the HTTP Server (original) support, and is then extended by allowing the user to specify one of the following values: Off, Hourly, Daily, Weekly, or Monthly. The directive that provides log rollover support is LogCycle.

Another feature that was recently introduced to better manage HTTP server log files is an *automatic expiration management*. Administrators can now choose to have the HTTP server delete log files that are expired. For more information, see 13.2.3, “Server logs” on page 331.

1.1.15 Domino plug-in

With Domino 6 for iSeries, clients can take advantage of the HTTP Server (powered by Apache) to forward HTTP traffic to their Domino 6 servers. This plug-in supports the HTTP Server (powered by Apache) at V5R3 of i5/OS, V5R2 of OS/400, and was sent back by PTF to V5R1. See *IBM Lotus Domino 6 for iSeries Implementation*, SG24-6592, for details about the HTTP Server (powered by Apache).

For the latest information about Lotus Domino, see:

<http://www.ibm.com/servers/eserver/iseries/domino/>

1.1.16 WebSphere Application Server plug-in

The HTTP Server (powered by Apache) handles static content, CGI program invocations, and proprietary plug-ins. The WebSphere Application Server run-time environment plugs into IBM HTTP Server for iSeries using plug-in APIs. This extends the support of the HTTP Server to include an implementation of the Java 2 Platform Enterprise Edition (J2EE) specification from Sun Microsystems. See 9.1, “Web application servers for the iSeries server” on page 193, for details.

1.1.17 Apache Software Foundation's Jakarta Tomcat

The HTTP Server (powered by Apache) includes an industry-standard Java servlet and JavaServer Pages (JSP) container, based on technology from the Apache Software Foundation's Jakarta Tomcat open source code base. Lightweight and easy-to-use software extends the HTTP Server (powered by Apache) server and is compliant with the Java Servlet 2.2 and JavaServer Pages 1.1 specifications from SUN Microsystems.

Apache Software Foundation's Jakarta Tomcat for iSeries support can be used as a simple starting point for business partners and customers interested in learning about or piloting Java servlet and JSP applications. This support is based upon Version 3.2.4 of the Jakarta Tomcat specification.

For more information, refer to 9.2, “Apache Software Foundation’s Jakarta Tomcat on iSeries” on page 197, and the iSeries HTTP server Web page at:

<http://www.ibm.com/eserver/iseries/software/http>

In addition, Appendix B, “Bringing Tomcat Version 5.5 to your iSeries server” on page 409, provides an example that shows how to bring Tomcat Version 4.1 on the iSeries server. This is not supported directly by IBM.

1.1.18 Apache Portable Runtime and modules

The design of the Apache HTTP server defines *modules*. Modules are operating system objects that can be dynamically linked and loaded to extend the base nature of the Apache HTTP server. Depending on the operating system, this is similar to:

- ▶ Microsoft Windows® Dynamic Link Libraries (DLL)
- ▶ UNIX shared object libraries
- ▶ OS/400 Integrated Language Environment (ILE) Service Programs

In this way, the Apache modules provide a way to extend a server’s function. Functions commonly added by optional modules include:

- ▶ Authentication
- ▶ Encryption
- ▶ Application support
- ▶ Logging
- ▶ Support for different content types
- ▶ Diagnostics
- ▶ Compression

You can extend the core functionality of the HTTP Server (powered by Apache) by writing your own modules or porting other modules to the iSeries as demonstrated in Chapter 12, “Apache Portable Runtime: Extending your core functionality” on page 311.

1.1.19 Support for the TRCTCPAPP command

The Trace TCP/IP Application (TRCTCPAPP) command can be used to trace the server, but only one instance at a time. It can be started while the server is running. See 13.2.5, “HTTP server trace” on page 341, for more information about how to use the TRCTCPAPP command.

Note: The old -vv (very verbose) still works at startup much like the original server (and -vi and -ve, which stand for informational and error tracing, respectively). You can use the Dump User Trace (DMPUSRTRC) and Display Physical File Member (DSPPFM) commands to see the results, but we recommend that you use the TRCTCPAPP trace method.

1.1.20 Collection Services performance data

The HTTP Server (powered by Apache) supports collection services and provides performance data that is specific to the HTTP server. It uses a feature that was introduced in V5R2 from Collection Services for user-defined categories and probes. This support allows IBM products and customer applications to capture application unique performance data along with the system data already provided by collection services.

Note: The HTTP Server (powered by Apache) does not support the `mod_status` module's `/status` function. Instead IBM has chosen to write performance-related information to collection services. For more information, see 13.2.6, “Collection Services performance data” on page 345.

1.1.21 Real-time server statistics

Real time server statistics provide information about HTTP Server (powered by Apache) performance. You can view server statistics using the Real Time Server Statistics tool that is available through the IBM Web Administration for iSeries interface. You can view only statistics for running HTTP Server (powered by Apache). Data is collected from the primary server job only. The statistics are real time and can help you to see how your server performs.

For example, you see information about proxy requests that were served from cache or retrieved from disk. In this example, the information can help you determine whether your proxy cache setup actually meets the user requests coming in to your server.

This function is available for OS/400 V5R1 and V5R2 via PTFs and in i5/OS. For more information, see 10.8, “Real Time Server Statistics” on page 301.

1.1.22 Triggered Cache Manager

Triggered Cache Manager (TCM) provides a mechanism to manage dynamically-generated Web pages. TCM is a separate server that can be used in conjunction with the HTTP server to allow a Web designer to build dynamic pages. It only updates the cache when the underlying data changes, thereby improving the performance of a Web site.

See 10.5, “Triggered Cache Manager” on page 259, for more information and a configuration example.

1.1.23 Fast Response Cache Accelerator

Working with the HTTP Server (powered by Apache), FRCA provides a cache mechanism that dramatically improves the file serving performance on your iSeries server. FRCA operates below the Machine Interface (MI) and therefore eliminates much of the overhead that is involved in switching to above MI threads. This enables FRCA to accelerate the delivery of an individual file found in its cache and reduce the amount of central processing unit (CPU) needed to handle the request (as compared to HTTP Server (powered by Apache)). FRCA can handle both a static file caching and a dynamic reverse proxy caching.

For more information, see 10.6, “Fast Response Cache Accelerator” on page 281.

1.1.24 Compression

The Apache module `mod_deflate` is a powerful tool that allows you to compress, by configuration, files that are being served from your HTTP Server (powered by Apache). `mod_deflate` is Apache's open source equivalent to `mod_gzip`. Compressing the data that is being sent by your HTTP Server (powered by Apache) can dramatically save on network delays due to bandwidth restrictions. The data is decompressed at the remote client's Web browser. `mod_deflate` is particularly useful in networks where individual links are saturated with traffic or the end-user is connected via modem. Of course, the compression of the data takes additional resources on both the server and client, so you must use care with this powerful module.

As an anecdotal example, the /index.html home page that is served from our NetObjects Fusion-generated sample Web application used in this redbook is compressed by mod_deflate from 10867 to 2002 bytes.

The initial compression support for the HTTP Server (powered by Apache) could be only configured manually by adding the corresponding directives to the HTTP server configuration file. Recent enhancements also provide configuration support via the IBM Web Administration for iSeries interface. See 10.4, “Increasing throughput with compression” on page 240, for more information and a configuration example.

1.1.25 Highly available HTTP server

If Web serving is a critical aspect of your business, you may want high availability for your Web server environment. A highly available Web server takes advantage of iSeries clustering technology and makes it possible to build a highly available Web site. This improves the availability of business-critical Web applications built with CGI programs.

See Chapter 14, “High availability” on page 355, for more information. You can find an example of this support in 14.2, “A working primary or backup with takeover IP model” on page 359.

1.1.26 Support for IASPs

An IASP is a collection of disk units that you can bring online or take offline, independent of the rest of the storage on a system. Each IASP contains all of the necessary system information associated with the data it contains. While the system is active, you can take the IASP offline, bring it online, or switch between systems. IASPs may contain:

- ▶ One or more user-defined file systems
- ▶ One or more external libraries

This feature was fully tested with the V5R2 delivery of the HTTP Server (powered by Apache).

1.1.27 Asynchronous I/O

The HTTP Server (powered by Apache) processes communications requests asynchronously. In this asynchronous input/output (I/O) model, threads are only involved in processing when there is work to be done. Threads are dispatched to perform work as required. When not performing work, the threads are returned to a pool of available threads making the server process more efficient and improving performance by better using the thread resources. Asynchronous I/O also makes the server more scalable to support a high number of users especially when combined with persistent connections.

For more information, see 10.2.1, “Threads and asynchronous I/O” on page 228.

1.1.28 Denial of service

The denial of service configuration directives are equally performance settings and a security. These directives allow you to identify, based on the data frame size, the possibility of an attack. The HTTP server may identify an attack because the frame size differs from the one it expects. Although this setting impacts the server performance as each request is tracked, it allows you to prevent a more dangerous performance degradation when dealing with a type of attack that may intentionally slow down or even completely paralyze your server.

For more information, see 10.2.7, “Denial of service” on page 233.

1.1.29 Miscellaneous

In addition to these functions, the following functions are provided only in HTTP Server (powered by Apache):

- ▶ Headers control: Has the ability to control headers, expires, and other headers
- ▶ More customization of directory listings
- ▶ Automatic restart of multi-threaded child job monitored by parent job
- ▶ Configuration file support in thread safe integrated file systems (not just QSYS.LIB)

1.2 For more information

For more information, see the IBM HTTP Server for iSeries Web site. This site is the center of much of the information related to the HTTP Server (powered by Apache). If you are reading this redbook, then you should bookmark this Web site in your Web browser:

<http://www.ibm.com/servers/eserver/series/software/http/>



From zero to powered by Apache

This chapter explains how to install the HTTP Server (powered by Apache) and get a basic HTTP server configuration up and running on your iSeries server. All the examples used in this IBM Redbook are based on this configuration.

Certain portions of this chapter are optional and they are identified as so. The amount of time it takes you to get to your first HTTP Server (powered by Apache) depends on where you are starting from and the options you choose along the way.

2.1 Before you start

This section can help you identify the software components, upgrades, user profile authorities, and Web browser requirements that you need to fully exploit the examples used in this IBM Redbook. Carefully reviewing this section now will save you time later.

Note: In this IBM Redbook, when we refer to the @server i5 or iSeries primary operating system, we refer to OS/400, even though the operating system name has changed to i5/OS in V5R3. If it is not explicitly stated otherwise, the term OS/400 also refers to i5/OS.

2.1.1 Software

Maybe the most time consuming aspect of starting any new enterprise on any computer system today is getting all the software and the fixes for that software that you will need to install and run without any trouble. This section helps you identify, in advance, exactly what you need to get your first HTTP Server (powered by Apache) up and running.

Products and options to OS/400 and i5/OS

Review Table 2-1 for the mandatory and optional software used by your HTTP Server (powered by Apache). This redbook is based on the latest available software release V5R3.

Note: Most of the software Licensed Program Products (LPP) and OS/400 options mentioned in Table 2-1 are already available on the shipped media and are no cost with your purchase of OS/400 5722-SS1 and i5/OS. In fact, on new shipments of the LPP 5722-DG1, some of the other LPPs are preloaded on many iSeries servers.

Table 2-1 Mandatory and optional software for your HTTP Server (powered by Apache)

Name of product or option	Product or option number	Comment
These LPPs and OS/400 options are mandatory:		
IBM HTTP Server for iSeries	5722-DG1	The LPP is IBM HTTP Server for iSeries. See Table 2-2 on page 20 for a list of all the components in 5722-DG1.
TCP/IP Utilities	5722-TC1	This is a useful collection of Transmission Control Protocol/Internet Protocol (TCP/IP) applications including Telnet and File Transfer Protocol (FTP).
Java Developer Kit 1.3	5722-JV1 *Base and Option 5	Your HTTP Server (powered by Apache) requires this LPP for the administration graphical user interface (GUI), commonly referred to as the <i>admin GUI</i> .

Name of product or option	Product or option number	Comment
These LPPs and options are recommended (but optional):		
WebSphere Development ToolSet	5722-WDS and Option 51: Compiler - ILE C	This is needed if you plan to do any application development in ILE languages. Chapter 12, "Apache Portable Runtime: Extending your core functionality" on page 311, has a source example written in C.
Triggered Cache Manager (TCM)	Option 1 of 5722-DG1	This is needed only if you will be working with TCM.
OS/400 Portable Application Solutions Environment (OS/400 PASE)	Option 33 of OS/400	If you plan to implement Hypertext Preprocessor (PHP) or other Common Gateway Interface (CGI) applications running in OS/400 PASE, see Appendix A, "Bringing PHP to your iSeries server" on page 387.
HA Switchable Resources	Option 41 of OS/400	Use this option if you plan to implement Highly Available (HA) Web servers as demonstrated in Chapter 14, "High availability" on page 355.
These LPPs and OS/400 options are optional and only needed if you plan to work with digital certificates and Secure Sockets Layer (SSL) and Transport Layer Security (TLS):		
Digital Certificate Manager (DCM)	Option 34 of OS/400	Optional: To support the handling of digital certificates used by SSL and TLS for secure Web serving
Cryptographic Access Provider	5722-AC2 or 5722-AC3	<p>If you want to use SSL or TLS, you must install one of the IBM Cryptographic Access Provider products.</p> <p>In V5R2, 5722-AC2 was withdrawn. You can order this as a separate no-charge LPP.</p>

Name of product or option	Product or option number	Comment
These LPPs and OS/400 options are optional and only needed if you plan to work with Java servlet and JavaServer Page (JSP) programming with either Apache Software Foundation's Jakarta Tomcat or WebSphere Application Server:		
WebSphere Application Server, Advanced Edition	5733-WA4 or	WebSphere Application Server enables the HTTP Server (powered by Apache) to serve servlets and JSPs. It enables Extensible Markup Language (XML) document processing. Both WA4 and the IWE are chargeable LPPs. ASF's Jakarta Tomcat is a component of 5722-DG1 (see Table 2-2).
WebSphere Application Server, Advanced Single Server Edition	5733-WA4 or	
WebSphere Application Server, Express for iSeries Version 5.1	5722-E51 or	
WebSphere Application Server V5.0 Base and Network Deployment for iSeries	5733-WS5 or	
WebSphere Application Server, V5.1 Base and Network Deployment for iSeries	5733-W51	
Toolbox for Java	5722-JC1	We recommend this if you plan to program in Java.
Java Developer Kit (JDK)	5722-JV1 Options	Your HTTP Server (powered by Apache) requires this LPP *Base and Option 5 (JDK 1.3). If you plan to program in Java, you may need other options for other levels of the JDK.
Qshell Interpreter	Option 30 OS/400	This is useful for working with WebSphere Application Server.

In addition, LPP IBM HTTP Server for iSeries (5722-DG1) contains the components shown in Table 2-2.

Table 2-2 Components available in LPP 5722-DG1

Name of component	Option number	Comment
HTTP Server (original)	*Base	The original IBM HTTP Server (only until OS/400 V5R2)
HTTP Server (powered by Apache)	*Base	"You are here."
Net.Data	*Base	See 7.3, "Net.Data: A ready-made scripting tool" on page 161.
Webserver Search Engine and Web Crawler	*Base	See Chapter 11, "Getting started with Webserver Search Engine and Web Crawler" on page 307.
Apache Software Foundation's Jakarta Tomcat	*Base	A Java servlet and JSP application server based upon Tomcat version 3.2.4. See 9.2, "Apache Software Foundation's Jakarta Tomcat on iSeries" on page 197.
Highly Available HTTP Server	*Base	See Chapter 14, "High availability" on page 355.
Triggered Cache Manager	Option 1 of 5722-DG1	See 10.5, "Triggered Cache Manager" on page 259.

Important: V5R2 was the last release to support the HTTP Server (original). To be ready for the future, always use the HTTP Server (powered by Apache). For details, see:

<http://www-1.ibm.com/servers/eserver/series/software/http/news/sitenews.html>

Group PTFs

Since the HTTP Server (powered by Apache) is new and ever-changing, it is mandatory that you install the latest program temporary fixes (PTFs) for IBM HTTP Server for iSeries (5722-DG1) and other related products. Table 2-3 shows the products and PTFs that you must install on your iSeries server.

Table 2-3 Group PTF information for key iSeries products

Product	PTF description	Comments
For the latest available PTFs for the HTTP Server (powered by Apache), see this Web site and click PTFs and Support : http://www.ibm.com/servers/eserver/series/software/http/		
IBM HTTP Server for iSeries (5722-DG1) (V5R1)	Group PTF SF99156	You can display the current installed version of this V5R1 group PTF on the iSeries by entering the following command: DSPDTAARA QHTTPSVR/SF99156
IBM HTTP Server for iSeries (5722-DG1) (V5R2)	Group PTF SF99098	You can display the current installed version of this V5R2 group PTF on the iSeries by entering the command: WRKPTFGRP SF99098
IBM HTTP Server for iSeries (5722-DG1) (V5R3)	Group PTF SF99099	You can display the current installed version of this V5R3 group PTF on the iSeries by entering the command: WRKPTFGRP SF99099
You can find the latest available PTFs for JDK on the Web at: http://www-1.ibm.com/servers/eserver/support/series/index.html Click Fixes → Group PTFs → R510 or R520 or R530 and the appropriate Group PTF number.		
Java Developer Kit (5722-JV1) (V5R1)	Group PTF SF99069	You can display the current installed version of this V5R1 group PTF on the iSeries by entering the command: DSPDTAARA QJAVA/SF99069
Java Developer Kit (5722-JV1) (V5R2)	Group PTF SF99169	You can display the current installed version of this V5R2 group PTF on the iSeries by entering the command: WRKPTFGRP SF99169
Java Developer Kit (5722-JV1) (V5R3)	Group PTF SF99269	You can display the current installed version of this V5R3 group PTF on the iSeries by entering the command: WRKPTFGRP SF99269

Keep in mind that group PTFs are updated periodically with the latest PTFs. The group PTF number does not change. During an update, any additional PTFs that impact the HTTP Server (powered by Apache) are added to the group PTF.

Tip: If you plan to install and operate any product of the WebSphere family, ensure that you install the latest group PTFs for the appropriate license program. The group PTFs for WebSphere also include group PTFs for the HTTP Server (powered by Apache) and JDK. For more information about WebSphere group PTFs, see:

<http://www.ibm.com/servers/eserver/series/software/websphere>

Use care and install every shipped group PTF if you are working with WebSphere and the HTTP Server (powered by Apache), because the level of all involved licensed programs should match in order to work properly.

2.1.2 User profile authorities

To use the GUI for configuration and administration, a valid iSeries user profile and password are required. You must have the following authorities to perform configuration and administration tasks:

Tip: We recommend that you *do not* use QSECOFR. Use a user profile with the appropriate authority.

- ▶ Your user profile must have *IOSYSCFG authority.
- ▶ Your user profile must have *CHANGE authority to the library object QUSRSYS.

The following file objects require *ALL authority:

- ▶ QUSRSYS/QATMHINSTA
- ▶ QUSRSYS/QATMHINSTC

The following command objects require *USE authority:

- ▶ CRTVLDL
- ▶ STRTCPSVR
- ▶ ENDTCPSPVR

QTMHHTTP is the default user profile of the HTTP Server. QTMHHTTP1 is the default profile that the HTTP Server uses when running CGI programs. The HTTP Server profile must have *RWX authority to the directory path where the HTTP Server (powered by Apache) configuration files are stored. The default path is `/www/servername/`, where *servername* is the name of the HTTP server instance.

The HTTP Server profile must have access to the directory path where the log files are stored. Fully consider the security of the log files. The path of the log files should only be accessible by the appropriate user profiles.

2.1.3 Web browser

The admin GUI for both the original and powered by Apache configuration is served from an HTTP Server (powered by Apache).

HTTP Server (powered by Apache) is configured using a client Web browser. To use the Configuration and Administration forms, you need a Web browser that supports:

- ▶ HTTP 1.0 or 1.1 protocol
- ▶ Frames
- ▶ Java Script

Such browsers as Microsoft's Internet Explorer 5.5 or later and Netscape Navigator 4.75 work with the configuration and administration GUI forms.

Tip: As you can see in this IBM Redbook, most configurations are tested and all windows have been captured with Microsoft's Internet Explorer. We recommend that you use the same browser.

To view the log reports generated by the HTTP Server, you must use a browser that supports Java Virtual Machine 1.1.5 or later.

2.2 Software installation

The installation of software on the iSeries can be split into two pieces. First, install or make sure that you have installed the required LPPs and OS/400 options. Then, update those products with the latest PTFs available.

You can perform an optional third step now or later depending on how closely you want to follow the how-to steps demonstrated in this IBM Redbook. This optional step explained in 2.2.3, "Installing the ITSO example Web application (optional)" on page 24, required you to download a simple Web application from the International Technical Support Organization (ITSO) Internet FTP server and install it on your iSeries server.

2.2.1 Installing LPPs and OS/400 options

Follow these steps to install the different LPPs and OS/400 options that you determined that you want to use after reviewing Table 2-1 on page 18:

1. Insert the installation media into your iSeries server.
2. At the OS/400 command line, type:

```
GO LICPGM
```


Press Enter.
3. Select option 11 (Install licensed programs) on the Work with Licensed Programs display to see a list of licensed programs that you can install on your iSeries server.
4. Select and install desired LPPs and OS/400 options.

Tip: See the Software Installation Guide at the iSeries Information Center, under OS/400 and related software, for help with licensed program installation. You can find the Information Center on the Web at:

<http://www.ibm.com/series/infocenter>

5. To verify your installation, select option 10 on the Work with Licensed Programs (GO LICPGM) menu. Make sure that the installation status for the options you installed is *COMPATIBLE for DG1 and either *COMPATIBLE or *INSTALLED for your other products.
6. Although this is not really part of the installation process, make sure that the system value Shared Memory Control (QSHRMEMCTL) value is set to 1 (Allowed). To display (and change, if necessary) the current system value setting on your iSeries server, use the Work with System Values (WRKSYSVAL) command:

```
WRKSYSVAL SYSVAL(QSHRMEMCTL)
```

2.2.2 Installing PTFs

To ensure a smooth test of the HTTP Server (powered by Apache) and to guarantee that you have all the functions available, you must load the latest PTFs. Table 2-3 on page 21 identifies the key group PTFs that you need to order and install on your iSeries server.

Tip: See the Software Installation Guide on the iSeries Information Center, under OS/400 and related software, for help with receiving and installing PTFs on your iSeries server. You can find it on the Web at:

<http://www.ibm.com/iseries/infocenter>

2.2.3 Installing the ITSO example Web application (optional)

To establish a quick, simple, and ready to use Web site for this IBM Redbook, we used NetObjects' Fusion 5.0. You can find the NetObjects Web site at:

<http://www.netobjects.com/>

Our goal was to have a Web site that had multiple layers of integrated file system (IFS) hierarchy that could then be configured in many different ways showing you the powerful Apache directives at work.

If you want to closely follow the how-to steps demonstrated in this IBM Redbook, be sure to download the examples as explained in Appendix D, "Additional material" on page 421.

2.3 Testing the HTTP Server (powered by Apache) installation

Now that you made sure you have all the components and fixes to those components that you need to work with the HTTP Server (powered by Apache), test them by creating and starting your first server.

2.3.1 Your first HTTP Server (powered by Apache) via a wizard

The HTTP Server (powered by Apache) comes complete with a GUI that is unique to the iSeries server. This GUI has dramatically changed since the introduction of V5R2 and has become user friendly and comfortable. It is also available at V5R1 if you installed the latest Group PTF for 5722-DG1.

Tip: The HTTP Server (powered by Apache) ships with a basic server instance by the name APACHEDFT. Any time you create a new server on the iSeries server, *always* use the GUI wizard even if later you plan to manually edit the configuration file.

Creating the server

You use the Create HTTP Server wizard to quickly create your first brand new HTTP Server (powered by Apache):

1. Start the HTTP Administration server (*ADMIN). Using iSeries Navigator, expand **Network** → **Servers** → **TCP/IP** and right-click **HTTP Administration**. Click **Start**.

Or, from the iSeries command line, enter:

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
```


2. Verify that the *ADMIN server is up and running. Using iSeries Navigator, expand **Network** → **Servers** → **TCP/IP**. On the right, the HTTP Administration server should have a status of *Started*.

Or, from the iSeries command line, enter:

```
WRKACTJOB SBS(QHTTSPVR)
```

As soon as all ADMIN server jobs reach SIGW status as shown in Figure 2-1, you are ready. Note that the startup time may take a minute.

Work with Active Jobs						AS20
						01/09/02 15:32:53
CPU %:	.8	Elapsed time:	01:08:50	Active jobs:	190	
Type options, press Enter.						
2=Change 3=Hold 4=End 5=Work with 6=Release 7=Display message						
8=Work with spooled files 13=Disconnect ...						
Opt	Subsystem/Job	User	Type	CPU %	Function	Status
	QHTTSPVR	QSYS	SBS	0,0		DEQW
	ADMIN	QTMHHTTP	BCH	0,0	PGM-QZHBHTTP	SIGW
	ADMIN	QTMHHTTP	BCI	0,0	PGM-QZSRLOG	SIGW
	ADMIN	QTMHHTTP	BCI	0,1	PGM-QZSRHTTP	SIGW
						Bottom
Parameters or command						
==>						
F3=Exit F5=Refresh F7=Find F10=Restart statistics						
F11=Display elapsed data F12=Cancel F23=More options F24=More keys						

Figure 2-1 WRKACTJOB SBS(QHTTSPVR) showing the ADMIN server ready and waiting for work

3. Open a Web browser and enter:

`http://your.server.name:2001`

Here *your.server.name* is the name or the IP address of your iSeries server.

4. Enter your user ID and password in the window shown in Figure 2-2.

Figure 2-2 Entering the OS/400 user ID and password when prompted

You are greeted by the iSeries Tasks page (Figure 2-3) which can be different, depending on the optional LPPs that are installed on your iSeries server.

5. On the iSeries Tasks page (Figure 2-3), click **IBM Web Administration for iSeries**.



Figure 2-3 iSeries Tasks page for your iSeries

6. Click the **Manage** tab and then the **All Servers** subtab. As shown in Figure 2-4, you are presented with a list of all the configured HTTP instances.

An improvement to the administration interface is that you can now easily identify which servers are running and on which ports and addresses they are listening. From here, you can Start, Stop, Restart, Delete, Rename, or Manage your servers. You can click any existing server to quickly move to the configuration panel of the server.

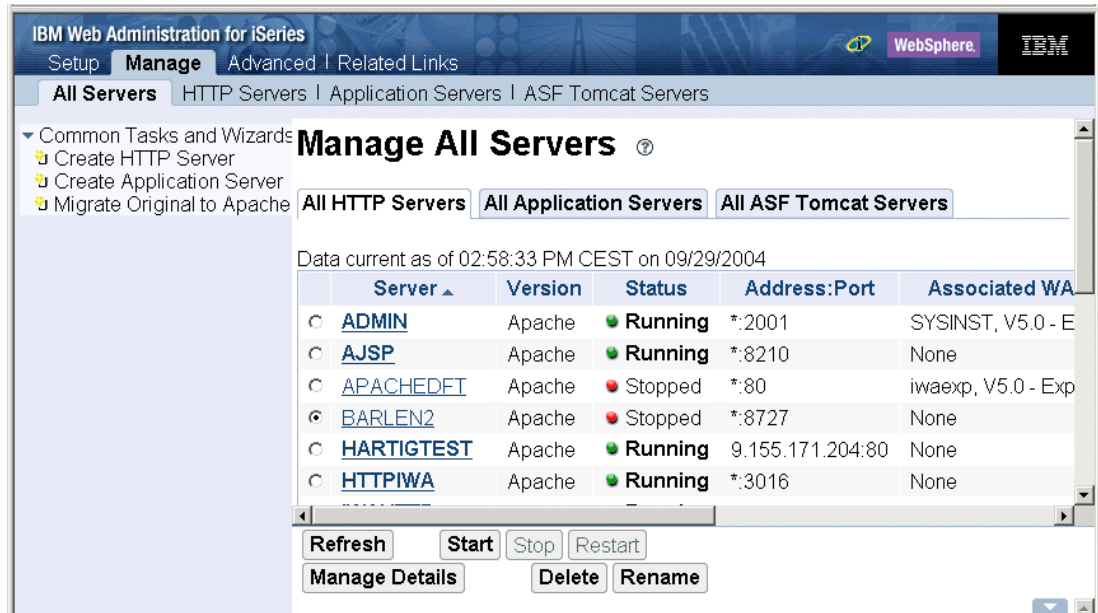


Figure 2-4 Managing all servers on one page

7. Start the Create HTTP Server wizard. In the left pane of the page, click **Create HTTP Server**. You see the first welcome panel for the Create HTTP Server wizard (Figure 2-5) in the right panel.

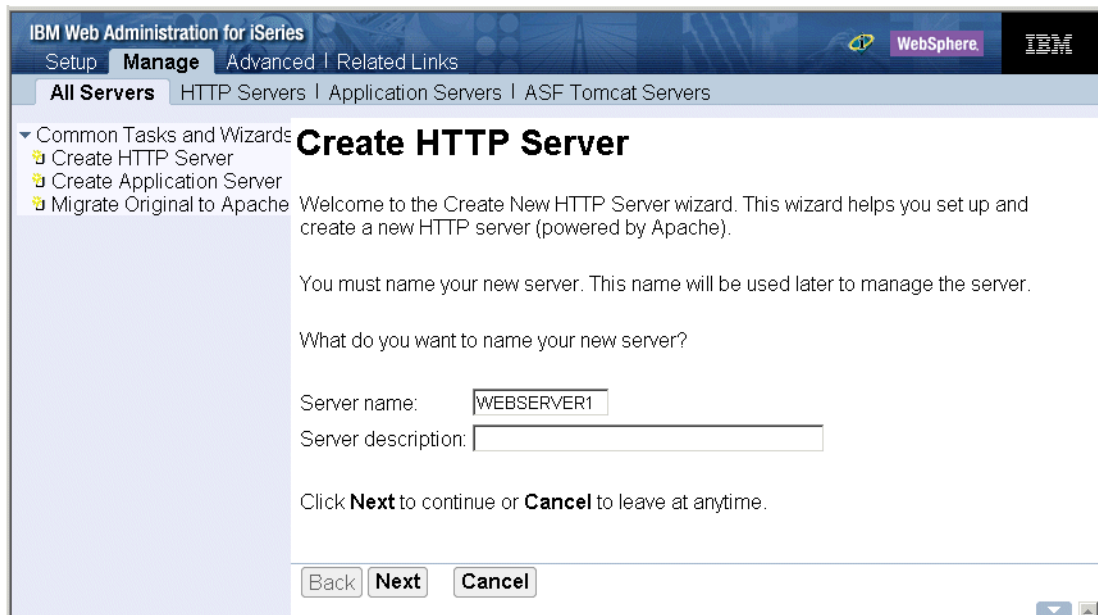


Figure 2-5 Create New HTTP Server: Welcome panel

Wizards are nice tools that just about guarantee that if you answer all the questions correctly, the resulting configuration will work. The only problem with a wizard is that sometimes, right in the middle, it asks you a question to which you do not know the answer. Table 2-4 can help with this situation. It contains the questions and our ITSO Rochester answers for the Create HTTP Server wizard. You can also write down your own answers if your first HTTP Server (powered by Apache) is different than ours.

Table 2-4 Questions asked by the Create HTTP Server wizard and ITSO Rochester's answers

Create HTTP Server wizard question	Answer
HTTP server name	ITSONEW
Migrate Original server configuration? (Yes/No) This question appears only when you have an existing original server configuration.	No
Server root. This is the base directory for your HTTP server. Within this directory, the wizard creates subdirectories for your logs and configuration information. The default is /www/webserver.	/www/ITSONEW
Document root. This is the directory from which your documents will be served by your HTTP server. The default is /www/webserver/htdocs.	/www/ITSONEW/htdocs
Listen on IP address and Port. The default is All IP addresses and Port 80.	All IP addresses
Port 8022 (or something unique for testing)	8022
Logging: Access log file or only Error log. You can find more information about logging in 13.2.3, "Server logs" on page 331.	No Access log
Log maintenance	Delete based upon age 7 days

Tip: If you run the Create HTTP Server wizard on a V5R1 or V5R2 system, you are asked an additional question regarding the type of server: original or (powered by Apache). Since i5/OS V5R3 does not support the original server anymore, this question is not shown.

- The Create HTTP Server wizard asks a series of questions as shown in Table 2-4. Answer each question and click **Next** to move onto the next question.

- In the end, you should see a confirmation panel that looks something like the example in Figure 2-6. Verify all your choices. If you need to correct something, click **Back**. Otherwise, click **Finish** to create your HTTP Server (powered by Apache).

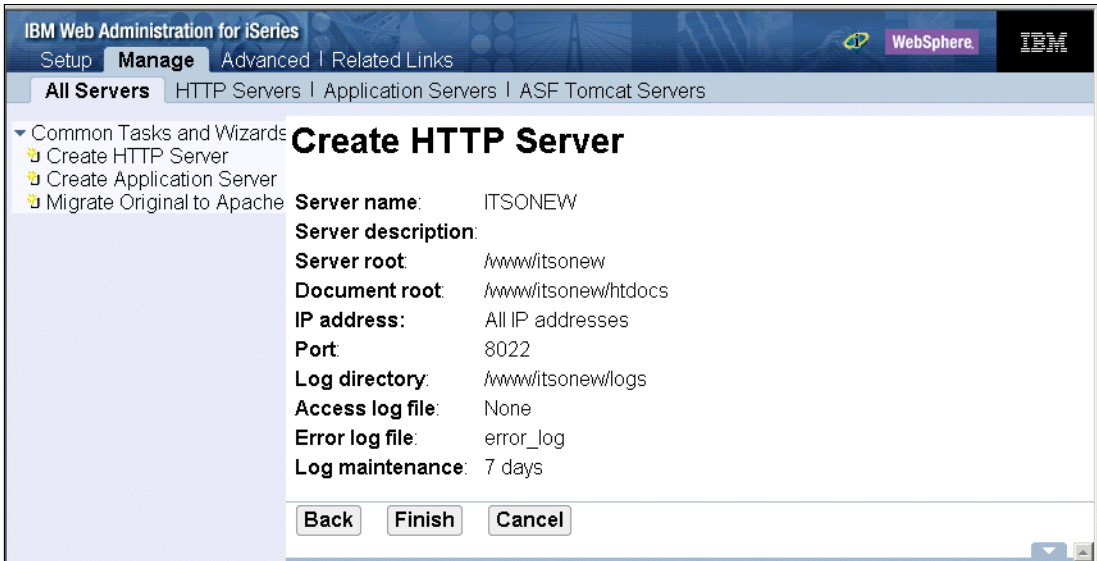


Figure 2-6 Create New HTTP Server: Confirmation panel

Note: We recommend that you *always* use the GUI to create new HTTP servers.

When the wizard has finished setting up the new instance, you see a new page that allows you to further configure your new server instance. See Figure 2-7.

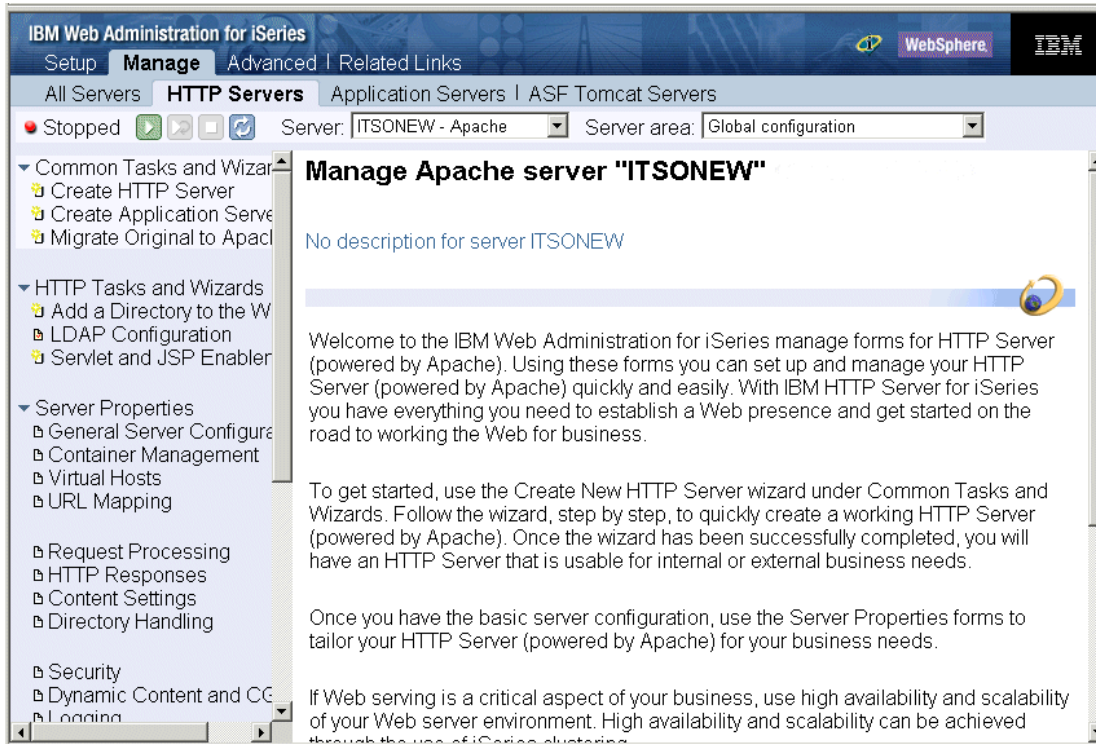


Figure 2-7 Managing details for the new server instance

Starting the server

Now that you have created your HTTP Server (powered by Apache), it is time to start it.

1. Make sure that you are still on the HTTP Servers tab, under the Manage tab, for your instance as shown in Figure 2-7.
2. The left frame changes and additionally shows the manage buttons (Figure 2-8). Click the green **Run** button (circled in Figure 2-8) and the server starts. You can alternatively start the server by entering the command:

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(ITSONEW)
```

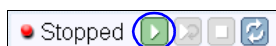


Figure 2-8 Buttons to manage your server

3. To verify that your ITSONEW server is up and running, click the **Refresh** button (circled in Figure 2-9). Or you can enter the following command:

```
WRKACTJOB SBS(QHTTPSVR)
```

Tip: Clicking the Refresh button a few times may be a good habit to ensure that the server *stays* started. This is similar to the nervous habit of old S/38, AS/400, and iSeries administrators repeatedly pressing PF5 (refresh) on a 5250 green screen. Sometimes, due to TCP/IP port conflicts or errors in the configuration file, the server terminates immediately. If this is the case, see Chapter 13, “Problem determination: When things do not go as planned” on page 323.

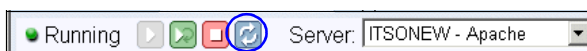


Figure 2-9 Your HTTP server ITSONEW is running

Testing the server

The final step is to simply test. Open your Web browser and enter:

`http://your.server.name:port`

Here *your.server.name* is the name or the IP address of your iSeries server, and *port* is the TCP/IP port on which your HTTP server is listening. In our example, we use the URL:

`http://hamts810:8022`

You now see the default home page (Figure 2-10), which is stored in the /www/ITSONEW/htdocs IFS directory and is named index.html.

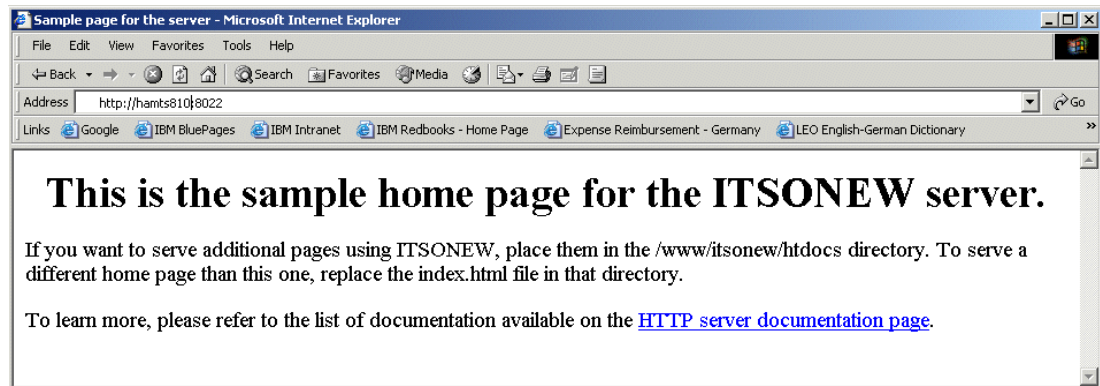


Figure 2-10 Default index.html created automatically by the Create HTTP Server wizard

You have now created and are running an HTTP Server (powered by Apache) on your iSeries server.



The new GUI: IBM Web Administration for iSeries

The graphical user interface (GUI) for the administration instance dramatically changed in V5R2 and in V5R1 (through a group PTF). The Rochester developers worked with the feedback of many iSeries clients to design and create a new look and feel to the iSeries administration GUI. New functionalities were added, and now, managing all of your servers has never been easier. See 3.3, “Tabbed pages for easy navigation” on page 36, for details.

Unlike the previous edition of this redbook, this chapter describes only the new look-and-feel that came to the administration GUI with PTFs for OS/400 V5R1 and V5R2 in December 2003, which is now in i5/OS V5R3 known as *IBM Web Administration for iSeries*. The administration GUI takes on more responsibility with the IBM WebSphere Application Server configuration. It allows you, the webmaster, to manage better the iSeries server.

This chapter gives you a general overview of the new functionalities. If you need more information about a particular configuration, refer to the appropriate chapter in this IBM Redbook.

3.1 Welcome page: iSeries Tasks page

The initial welcome page, or iSeries Tasks page, lists a wide range of applications that you can manage. To reach this page, follow these steps:

1. Make sure that TCP/IP is configured and started in your OS/400 or i5/OS partition.
2. Start the HTTP administrative server (as described in 1.1.2, “GUI configuration and administration” on page 5).
3. Open a Web browser and enter the following Uniform Resource Locator (URL):

`http://as400host.domain:2001`

The string *as400host.domain* stands for any IP address or host name that you can use to reach the OS/400 or i5/OS partition from your workstation.

4. You are prompted to enter your user profile and password (Figure 3-1). Sign on with your OS/400 user profile and password (refer to 2.1.2, “User profile authorities” on page 22, for more information about the required authority).

If the user profile and password prompt does not appear in the window shown in Figure 3-1, then most likely you need to start the Admin server. See 2.3.1, “Your first HTTP Server (powered by Apache) via a wizard” on page 24.

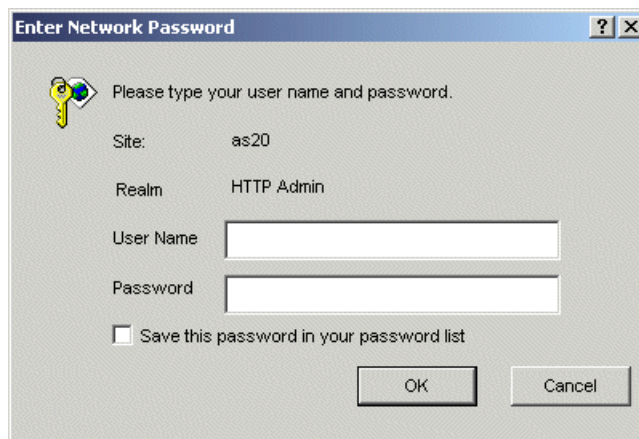


Figure 3-1 iSeries Tasks page: Login

5. The iSeries Tasks page opens (Figure 3-2). The number of tasks that you see depends on the installed licensed programs in this particular OS/400 or i5/OS partition on your iSeries server. Therefore you may or may not see all the same links as those shown in Figure 3-2. Click **IBM Web Administration for iSeries**.

Note: In OS/400 V5R2 and previous releases, this link was called IBM HTTP Server for iSeries.

You now go to the first page for the HTTP and application server administration.








Figure 3-2 iSeries Tasks page

Tip: Do you want to see this page in a different language? By default, the iSeries Tasks page appears in the language that is defined as the primary language for OS/400 or i5/OS. You can override the language setting by using the LANGID parameter of the user profile that signs on to the iSeries Tasks page. However, the pages that appear after you click the IBM Web Administration for iSeries link are displayed in the language that is defined in your browser settings. See 15.1, “Installing secondary languages” on page 374, for more information.

3.2 Header images to access information for help

The IBM Web Administration for iSeries user interface has several images in the header, or top most portion, of the GUI. These images as shown in Table 3-1 are hyperlinks to helpful information.

Table 3-1 Header images with links for more information

Header image	Description
	This is the image hyperlink to the iSeries Information Center entry page.
	This is the image hyperlink to the WebSphere Application Server Family Web page. This Web page contains information about WebSphere products, including support and service information.
	This is the image hyperlink to the IBM Web page where you can find information about all of IBM's products.
	This is the image hyperlink to the IBM HTTP Server for iSeries Web page. This Web page contains additional information about PTFs and support, developer documentation, and other topics.
	You can find the question mark icon on many property forms near one or more input fields. Clicking this icon points you to help information for this particular configuration parameter.

3.3 Tabbed pages for easy navigation

The IBM Web Administration for iSeries GUI consists of several Web pages and wizards. As you can see in Figure 3-3, four tabs are shown at the top of the page to guide you to an individual section. You can easily access a specific page by clicking the tabs or subtabs at the top of the page. The following sections look at the most important ones. The main task tabs on the top (referred to in the documentation as *tabs*) are:

- ▶ “Setup tab: Common tasks and wizards” on page 37
- ▶ “Manage tab” on page 37
- ▶ “Advanced tab” on page 51
- ▶ “Related links page” on page 57

If you click the Manage or Advanced tab, you see subtabs to further group the tasks:

- ▶ Under Manage:
 - All Servers (described on page 38)
 - HTTP Servers
 - Application Servers
 - ASF Tomcat Servers
- ▶ Under Advanced:
 - “Settings subtab” on page 51
 - “Internet Users and Groups subtab” on page 52
 - “Search Setup subtab” on page 55
 - “TCM subtab” on page 56

Note: When you access the IBM Web Administration for iSeries GUI again, it opens the tab and subtab that you last had open before you closed it.

3.3.1 Setup tab: Common tasks and wizards

The Setup tab contains the setup tasks for your servers as shown in Figure 3-3. Setup tasks include the common tasks and wizards for the IBM Web Administration for iSeries user interface. Simple “getting started” tasks and wizards are also available. Under Common Tasks and Wizards, the Setup page provides the ability to:

- ▶ Create a new HTTP server
- ▶ Create an application server, if you have the appropriate product installed
- ▶ Migrate an existing HTTP Server (original) to HTTP Server (powered by Apache)

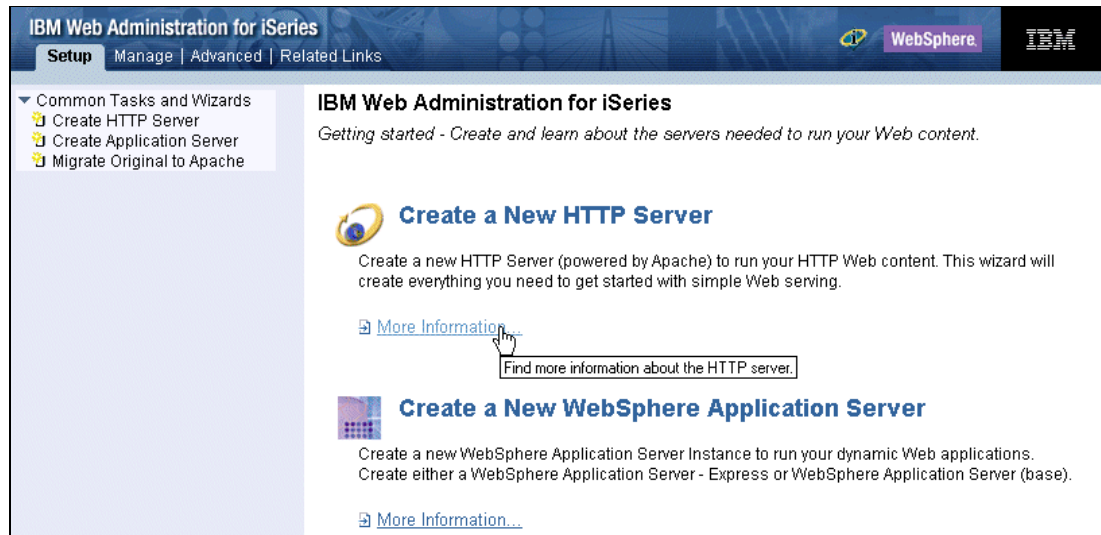


Figure 3-3 IBM Web Administration for iSeries: Setup page

To create an application server, WebSphere Application Server, Express 5.0 or higher, either the Express or the Base Edition needs to be installed on your system. Otherwise the second option is not shown in the list.

Note: The Common Tasks and Wizards option also appears on the Manage and Advanced tabs (except the TCM subtab).

In previous versions of the GUI, this page also offered the options to change the global server configuration, manage Internet users, or set up a search engine. These functions have now been moved to the Advanced tab.

For information about setup instructions, see 2.3.1, “Your first HTTP Server (powered by Apache) via a wizard” on page 24. To learn more about the migration wizard, see Chapter 8, “Migration from HTTP Server (original) to (powered by Apache)” on page 173.

If you plan to create or work with the WebSphere Application Server, Express, see the IBM Redbook *WebSphere Application Server V5 for iSeries: Installation, Configuration, and Administration*, SG24-6588.

3.3.2 Manage tab

The Manage tab contains the All Servers, HTTP Servers, Application Servers, and ASF Tomcat Servers subtabs. It allows you to access lists of all servers that are defined on your system or to set up new servers. You can control all servers from a single panel or choose a specific server to manage on your iSeries server.

Note: The HTTP Servers, Application Servers, and ASF Tomcat Servers subtabs, under the Manage tab at the top of the page, serve a different purpose than the All HTTP Servers, All Application Servers, and All ASF Tomcat Servers tabs, located under the All Servers tab, in the Manage All Servers panel.

Manage All Servers

The Manage page (Figure 3-4) is one of the most improved sections of the new GUI. It enables you to manage an individual server or to see the status of all your servers.

- ▶ All HTTP Server (powered by Apache) instances
- ▶ All HTTP Server (original) instances (only available when using OS/400 V5R2 or earlier)
- ▶ All WebSphere Application Server, Express Base or Express, instances for Version 5 or later
- ▶ All Apache Software Foundation (ASF) Tomcat servers

All HTTP Servers

The All Servers subtab (Figure 3-4) displays all of the currently configured servers on your iSeries server. It also provides you the ability to start, stop, restart, and configure your servers, as well as to monitor and manage the details.

The All HTTP Servers tab shows a table with all HTTP servers. It shows the ports and interfaces on which they listen and if a WebSphere Application Server instance is associated with the HTTP server. You can click any of the blue highlighted table headers (circled in Figure 3-4) to sort the table depending on the selected column.

IBM Web Administration for iSeries

Setup **Manage** Advanced | Related Links

All Servers HTTP Servers Application Servers ASF Tomcat Servers

Common Tasks and Wizards

- Create HTTP Server
- Create Application Server
- Migrate Original to Apache

Manage All Servers ?

All HTTP Servers All Application Servers All ASF Tomcat Servers

Data current as of 05:49:25 PM CET on 10/22/2004

Server	Version	Status	Address:Port	Associated WAS Instance	Description
ADMIN	Apache	Running	*:2001	None	Administration server
APACHEDEF	Apache	Stopped	*:80	default, V5.1 (base)	IBM supplied sample HTTP se
HARTIGTEST	Apache	Stopped	*:80	None	HTTP-Cluster Test
PHP	Apache	Stopped	*:8080	None	
QIPPSVR	Apache	Stopped	*:631	None	
THOMAS1	Apache	Stopped	*:8088 *:44388	None	
TOMITSQ1	Apache	Stopped	*:80 *:44306	None	Thomas' Web Server for Apac
WEBSERVER	Apache	Stopped	9.164.96.83:80	WASExp, V5.0 - Express	

Refresh Start Stop Restart

Manage Details Delete Rename

Figure 3-4 IBM Web Administration for iSeries: Manage All Servers

You can choose the server you want to work with. Simply select the server from the Server area drop-down box or select the proper radio button to mark your instance. If you use the radio button, you can choose from the following options:

Note: The Monitor Server option has been removed in V5R3, since it only applied to the HTTP Server (original). A similar function has been added to HTTP Server (powered by Apache) called *HTTP Server statistics*. See “Tools” on page 45 for more information.

If any of these operations fail, refer to Chapter 13, “Problem determination: When things do not go as planned” on page 323.

- ▶ **Refresh:** Click this button to refresh the status of all your servers.
- ▶ **Start:** This button enables you to start the selected server. The Server startup parameters allow you to add additional startup parameters, mostly used for debugging problems. See 13.2.7, “Other startup parameters” on page 351, for more information.
- ▶ **Stop:** Click this button to simply stop your server. It may be necessary to click the Refresh button to see the correct status.
- ▶ **Restart:** This is equal to the following OS/400 command:
`STRTCPSVR SERVER(*HTTP) RESTART(*HTTP) HTTPSVR(xxxxxxxxxx)`
- ▶ **Manage Details:** This option opens the main configuration page of the selected server. The same page opens when you select the server from the Server area list.
- ▶ **Delete:** This button offers the only opportunity where you can delete your server instance. When you click the Delete button, the server instance is removed from the list and you cannot retrieve the server status anymore. If the server you selected is running, it stops before the system deletes it. The system does not delete the server configuration that is associated with this server or the directory and its contents.
- ▶ **Rename:** Click this button if you have a reason to rename your server.

Tip: The area at the bottom of the page is where completion messages appear after any action. You can hide this area by clicking the green minimize button.

All Application Servers

You can now manage WebSphere Application Servers through the IBM Web Administration for iSeries (Figure 3-5). This enhancement was introduced in WebSphere Application Server, Express V5.0 and was later extended to WebSphere Application Server Base for Versions 5.0 and 5.1.

Although only a subset of all configuration parameters for application servers is available through this user interface, there are some advantages over using the WebSphere Administrative Console:

- ▶ You see the name, versions, status, ports and the descriptions of *all* application servers (as long as they are at version 5.0 or later and don’t include a Network Deployment server).
- ▶ You can use the table in Figure 3-5 to start an application server. In contrast, to use the WebSphere Administrative Console, the application server must already be active.

If you click the name of one of the application servers shown in the list, you can change or set many of the configuration parameters for this server, as described in “Application Servers subtab” on page 49.

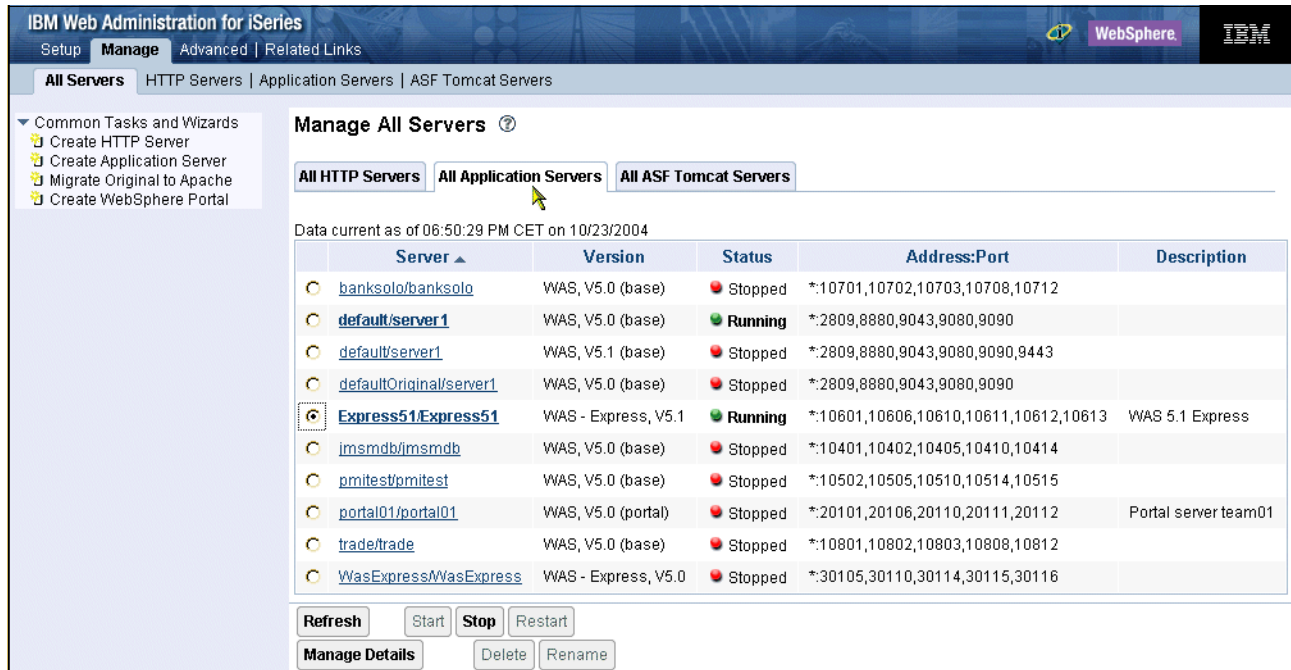


Figure 3-5 IBM Web Administration for iSeries: Managing all application servers

All ASF Tomcat Servers

Under the All ASF Tomcat Servers tab, you can also see and manage all Tomcat servers.

Managing each individual server

There are two ways to reach an overview page so you can manage or configure a particular server. The overview page describes the main server configuration tasks in the right panel and shows a collapsible tree of configuration tasks in the navigation bar on the left.

- ▶ Start with the All Servers tab as explained in “Manage All Servers” on page 38 and click the name of the server. If you want to manage an Application Server or ASF Tomcat Server, click the All Application Servers tab or the All ASF Tomcat Servers tab first to see a list of those server types.
- ▶ Click either the HTTP Servers, Application Servers, or ASF Tomcat Servers subtab. Then select the name of the server from the Server list under the subtabs (see Figure 3-6).

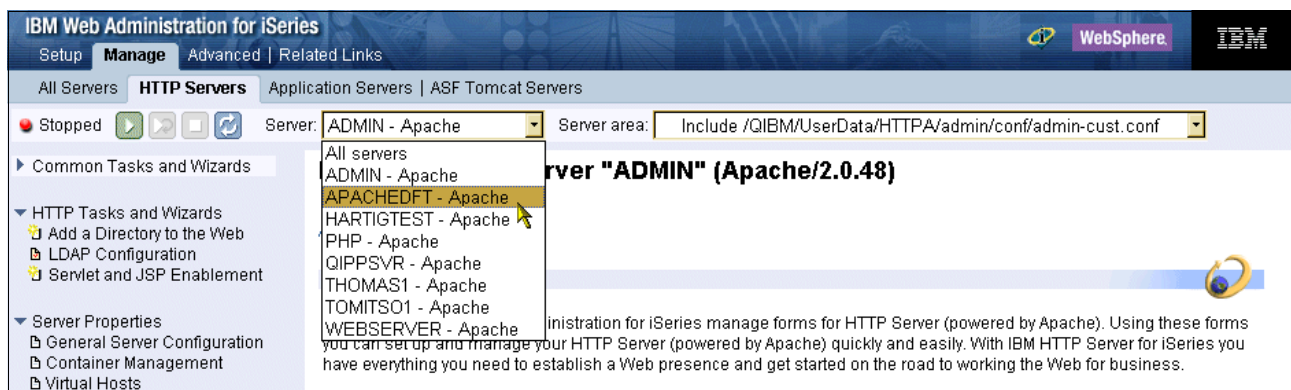


Figure 3-6 IBM Web Administration for iSeries: Selecting a server

HTTP Servers subtab

The HTTP Servers subtab allows you manage or configure a single HTTP server. Figure 3-7 shows a sample page of the APACHEDFT instance, which is the IBM supplied sample HTTP server (powered by Apache).



Figure 3-7 IBM Web Administration for iSeries: Managing your server

Server status




At the top of the page, under the subtabs and circled in Figure 3-7, you can see the status of your server. Depending on the actual status, the different buttons are active on the right as shown in Figure 3-8.





Figure 3-8 Server status: Running, Stopping, Stopped

The IBM Web Administration for iSeries user interface shows the current status of your servers under the subtabs at the top of the page. The status of the currently selected server is indicated by the icons listed in Table 3-2.

Table 3-2 Server states

Server state	Description
 Stopped	The server is currently stopped. The server is no longer available. The IP address and port number are not in use.
 Running	The server is currently running. The IP address and port number are in use.
 Stopping	The server is attempting to stop. The IP address and port number are still in use.

Server state	Description
 Creating	The server is being configured and created. The IP address and port number are not in use.
 Loading...	The IBM Web Administration for iSeries interface is loading the selected form, wizard, or Web browser frame.

Server and server area pull-down menus

To the right of the action buttons, you see the pull-down list to select a different server as shown in Figure 3-6 on page 40.

Another menu farther to the right allows you to select the server area. The Server area list contains the different *container areas*, such as <VirtualHost> or <Directory> for the HTTP Server (powered by Apache) configuration file.

Tasks, wizards, property forms, and tools

These options are shown in the left pane in Figure 3-7. These are the main wizards that help you to set up your server. Each subtab opens specific tasks, wizards, property forms, and tools that provide you the ability to configure and manage your server. Table 3-3 explains each option in greater detail.

Table 3-3 Tasks, wizards, property forms, and tools

Name	Description
Task	A task is guided property form that takes you through advanced configuration steps, but it is not a wizard. It groups property forms together for advanced configuration tasks.
Wizards	Wizards provide instructional steps that guide you through a series of advanced steps to accomplish a task. They cannot save your progress and must be completed to successfully update or create a server.
Property forms	Property forms have field values that may be set for specific configuration requirements. Each property form has help text to assist you in managing your servers.
Tools	Tools provide easy access to log files, the server configuration file, directive index, and real-time HTTP server statistics. They are useful for problem solving and server maintenance.

The IBM Web Administration for iSeries interface checks any changes you make for errors. It displays a message, below the forms (in the error window), detailing any errors.

Important: Do not resize the browser window during any operation due to a problem when using wizards. Otherwise, you are redirected to the page from which you started the wizard. At that point, there is no possibility to return to the wizard. You must start it again. An error message displays, indicating that a wizard is already running and the previous configuration data will be lost.

Server Properties

The collapsible tree in the left navigation bar is your main configuration tool. When you expand the subtree under Server Properties, you can define all of the operation environments for your HTTP Server (powered by Apache).

By clicking any of the configuration sections, the content of the panel on the right changes and shows the appropriate property form. Figure 3-9, shows the General Server Configuration form. For example, because the APACHEDFT instance is configured to listen on port 80 and bind to all interfaces, we disabled the instance from starting automatically by changing the Autostart parameter to No and clicking Apply.

The screenshot shows the 'General Server Configuration' form in the IBM Web Administration for iSeries. The left navigation pane is expanded to 'Server Properties' > 'General Server Configuration'. The main panel shows the 'General Settings' tab. The 'Autostart' dropdown is set to 'No'. The 'Associated WAS Instance' is 'default'. The 'Start all WebSphere application server(s) for the associated WAS instance when this HTTP server is started' dropdown is also set to 'No'. The 'Server root directory' is '/www/apachedft'. The 'Configuration file' is 'conf/httpd.conf'. The 'Document root' is '/www/apachedft/htdocs'. The 'Server name' section is empty. The 'Server IP addresses and ports to listen on' table shows two entries: 'All IP addresses' on port 80 and '*' on port 80, both with 'FRCA' set to 'Disabled'. The 'Tools' section at the bottom has 'OK', 'Apply', 'Cancel', and 'Preview' buttons.

	IP address	Port	FRCA
Example	All IP addresses	80	Disabled
	*	80	Disabled

Figure 3-9 IBM Web Administration for iSeries: General Server Configuration (Part 1 of 2)

The message area tells you that the configuration was updated successfully. You also see a message in this area if you encounter an error during your configuration.

Because of the larger size of this panel, you must scroll down to see the rest of the configuration parameters. Figure 3-10 shows the rest of this panel after we scrolled down.

Tip: While working with the IBM Web Administration for iSeries, you may want to use as much space of your display as possible. In Internet Explorer, you can do this by selecting View → Full Screen (F11) function. However, if you do this while a certain form is displayed, you return to the overview page (Figure 3-7 on page 41).

Document root: ?

Server name:

 Fully qualified server host name: ?

 Port: ?

Server IP addresses and ports to listen on: ?

	IP address	Port	FRCA
Example	All IP addresses	80	Disabled
<input type="radio"/>	*	80	Disabled

?

Number of threads to process requests: or... ?

DNS hostname lookups for logging, CGI, and SSI: ?

☒ Do not perform DNS lookups
☐ Perform DNS lookups
☐ Perform double-reverse DNS lookup

Follow symbolic links: ?

Follow symbolic links when target has same owner as the link: ?

Figure 3-10 HTTP Server Administration - General Server Configuration (Part 2 of 2)

You can add another port to your configuration. Simply click the Add button as indicated by the mouse pointer in Figure 3-10. Notice that this button is not visible in Figure 3-9.

Then a new row is added to the list of ports as shown in Figure 3-11.

Server IP addresses and ports to listen on: ?

	IP address	Port	FRCA
Example	All IP addresses	80	Disabled
<input type="radio"/>	*	80	Disabled
<input checked="" type="radio"/>	* <input type="text" value=""/> or... ?	<input type="text" value=""/>	<input type="text" value="Disabled"/> ?

Number of threads to process requests: or... ?

DNS hostname lookups for logging, CGI, and SSI: ?

☒ Do not perform DNS lookups
☐ Perform DNS lookups
☐ Perform double-reverse DNS lookup

Follow symbolic links: ?

Follow symbolic links when target has same owner as the link: ?

Figure 3-11 HTTP Server Administration: Adding a port to listen on

Then, you can click the Preview button to review the configuration as shown in Figure 3-12. The Preview button on this page lets you review your server configuration before you apply all the configurations. Notice (in the circled area in Figure 3-12) that the configuration changes are marked with a plus sign (+) if a configuration section is added.

If you remove a configuration directive, there is no indication in the configuration file. It is simply removed.



Figure 3-12 HTTP Server Administration: Preview Configuration File

Tools

Tools provide easy access to the server configuration file, directive index, and real-time HTTP server statistics. Tools are useful for problem solving and server maintenance. This section is located on the bottom of the navigation bar on the left side.

Depending on size and resolution of your display, if you do not see the Tools tree, you must scroll down, using the scroll bar on the right side of the left navigation pane, or collapse one more of the subtrees above it. The tools allow you to:

- ▶ Display the configuration file (see Figure 3-13)
- ▶ Edit the configuration file (see Figure 3-14 on page 47)

- View an index of all directives (see Figure 3-15 on page 48)
- View the real-time server statistics (see Figure 10-45 on page 302)

The Display the Configuration File Tool (Figure 3-13) allows you to see the raw content of the configuration file, which is usually conf/httpd.conf within the server root directory.

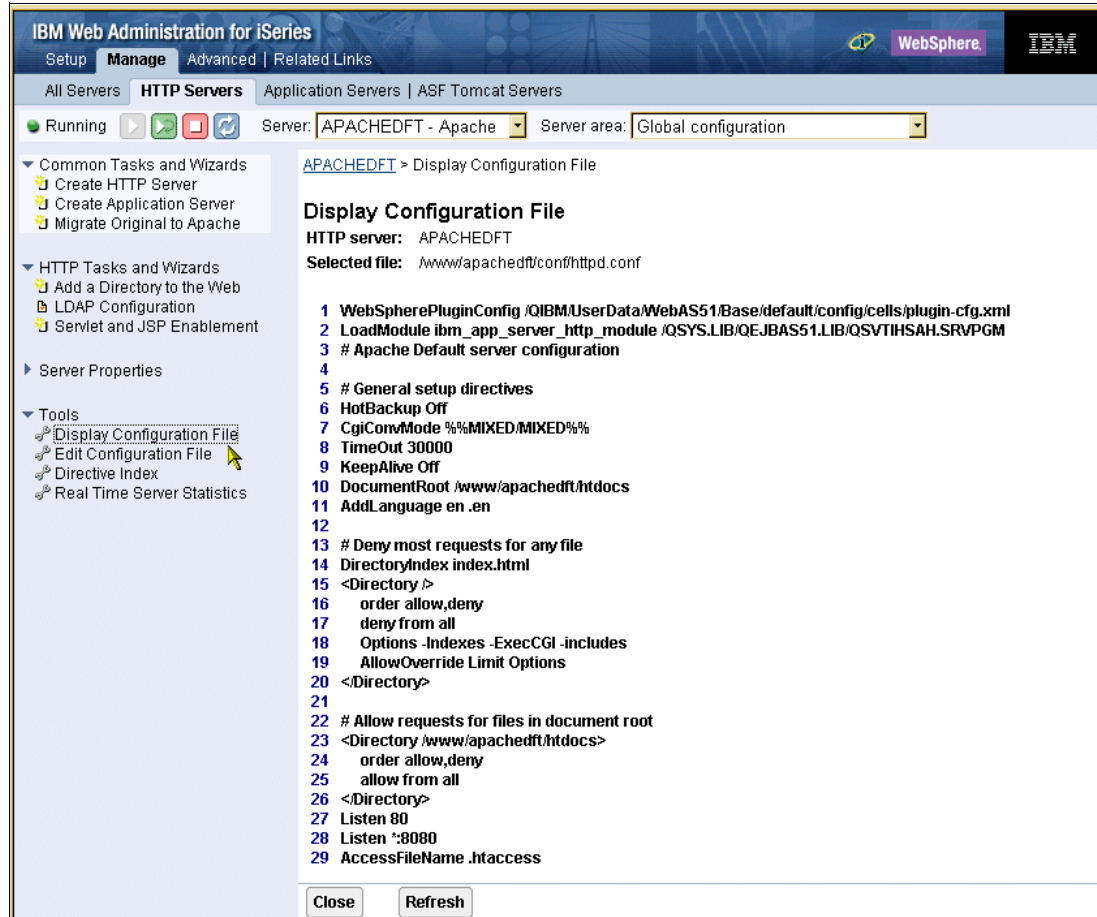


Figure 3-13 IBM Web Administration for iSeries: Display Configuration File

With the Edit Configuration File Tool, you can also edit the configuration file if you want to add your directives manually as shown in Figure 3-14.

Tip: Did you know that the httpd.conf file, created by the administration GUI, is in Unicode?

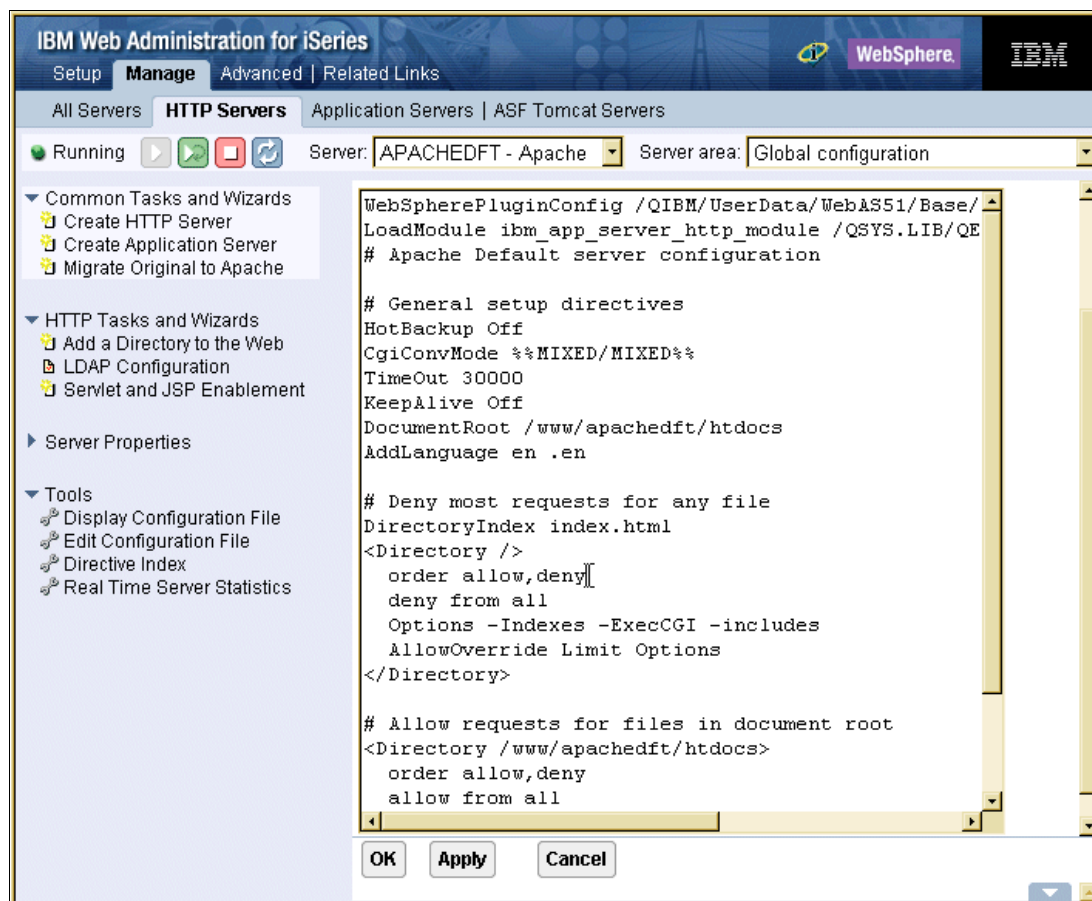


Figure 3-14 IBM Web Administration for iSeries: Edit Configuration File

Note: If you add a directive that is not supported via the Edit Configuration File utility and return to the Display Configuration File page, you see an error. This is also a good way to examine and start debugging problems. For more hints and tips, see Chapter 13, “Problem determination: When things do not go as planned” on page 323.

If you prefer to edit the configuration file directly, you can:

- ▶ Use the Edit Configuration File Tool, which is another part of IBM Web Administration for iSeries as shown in Figure 3-14.
- ▶ Use the Work with Object Links (WRKLNK) CL command to change into the directory, where the configuration file is placed. Then use option 2 to edit. Or you can simply use the Edit File (EDTF) CL command.
- ▶ Use a client system with a mapped directory to the iSeries integrated file system (IFS) and a tool that can edit Unicode files. If no such editor is available, copy the configuration (via Edit Configuration File) and paste it to an ASCII encoded file.

The Directive Index (Figure 3-15) is an alphabetical list of all directives for the HTTP Server (powered by Apache). It consists of cross-references to the place in the GUI where you can configure this directive.

Tip: If a directive is not valid in the currently selected context, you cannot click to access the form. You must first change the context in the Server area list in the upper right side of the panel.

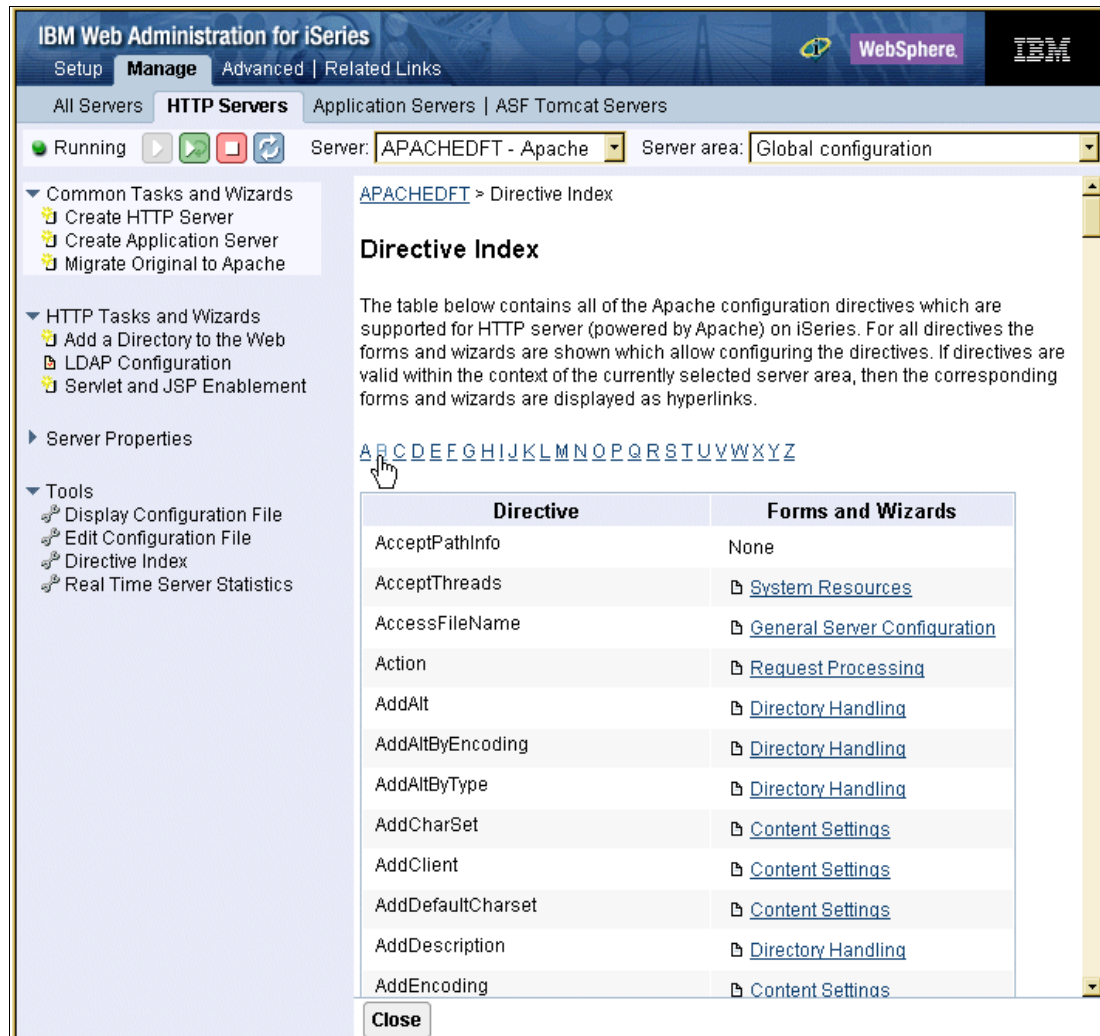


Figure 3-15 IBM Web Administration for iSeries: Directive Index

In our example, we search for the place to configure the directive `AccessFileName`. Viewing the Directive Index, we find it under General Server Configuration. Clicking this link opens the property form where you can change the configuration as shown in Figure 3-16.

APACHEDFT > Directive Index > General Server Configuration

General Server Configuration

General Settings
Welcome Pages
Configuration Includes
Advanced

OS/400 user profile to process requests:

Server CCSID:
or...

Client CCSID:

Include server description:
Server only

☐ URLs are case sensitive

How to build a self-referencing URL:

☐ Do not build a self-referencing URLs - use hostname and port supplied by client

☒ Use server name and port from configuration file

☐ Use hostname and port from a reverse DNS lookup

Access control file names:

	File name
Example	.htaccess
	.htaccess

Add

OK
Apply
Cancel
Preview

Figure 3-16 IBM Web Administration for iSeries: Directive Index, `AccessFileName`

Tip: For a certain input field in a property form, if you want to determine which directive the field controls, you can click the question mark icon as indicated by the mouse pointer in Figure 3-16.

Real Time Server Statistics

The Real Time Server Statistics form and tabs provide information about server performance. You can only view statistics for running servers. You may choose this form to be automatically refreshed every 10 or 30 seconds, or 1, 5 or 10 minutes, by selecting the option from the Refresh Interval list. The default is to refresh the data manually by clicking the Refresh button.

For complete details about Real Time Server Statistics, see 10.8, “Real Time Server Statistics” on page 301.

Application Servers subtab

WebSphere Administrative Console helps you to configure all functions of WebSphere Application Servers are Version 4 or later. Similar to the IBM Web Administration for iSeries, it is also a browser-based GUI. Unlike the IBM Web Administration for iSeries GUI, the WebSphere Administrative Console, in most cases, is only used to manage a single application server.

Attention: There is one exception. You can use the console of a Network Deployment (ND) server. In such a case, the ND server manages multiple Base (not Express) servers. However, the application servers must be defined that way and then can be managed only through the ND server.

The overview page for Manage Application Server – Express, in the example in Figure 3-17, shows the most important information about the server. It also provides links to guide you to key management and configuration forms and wizards.

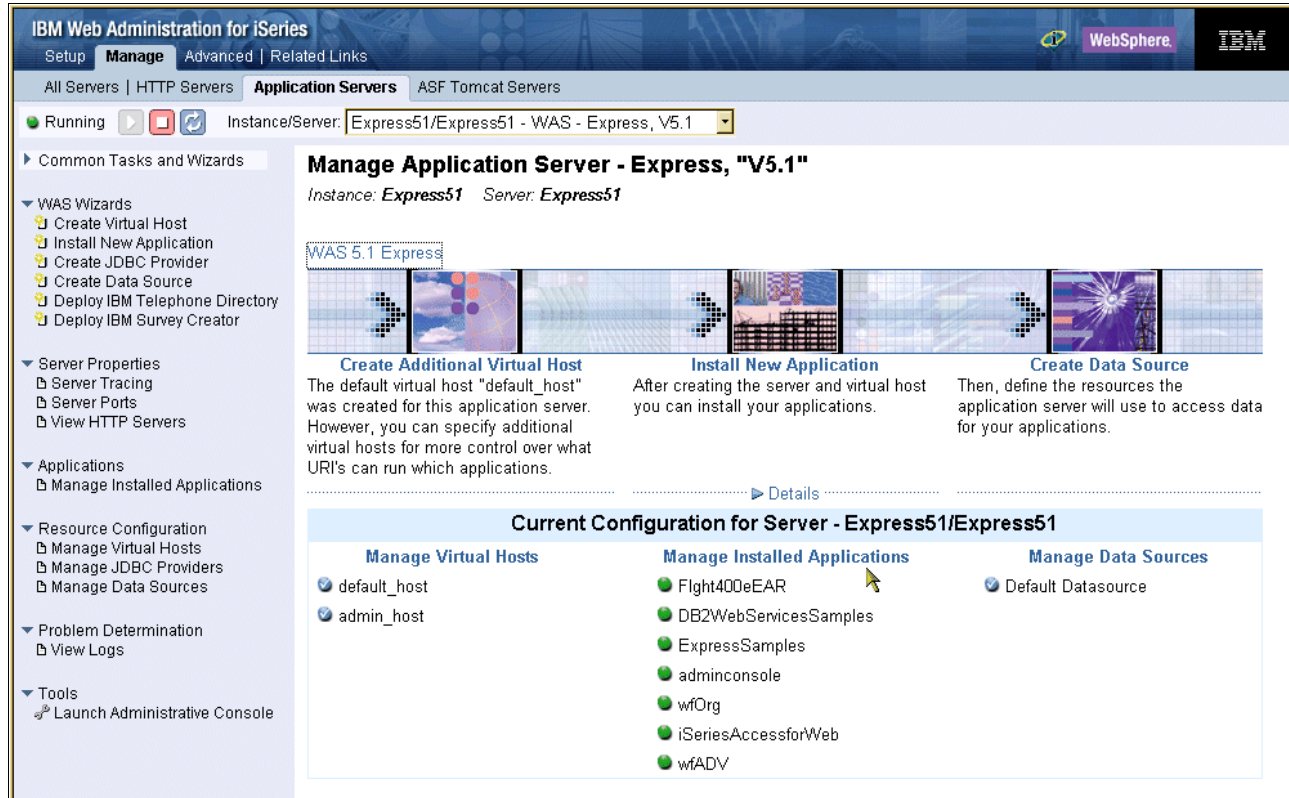


Figure 3-17 IBM Web Administration for iSeries: Manage Application Server

The top row of the Manage Application Server form looks similar to the one in the Manage HTTP server form, except that there is no Restart button or a Server Area list. However, the navigation bar in the left pane is different, except for the Common Tasks and Wizards links.

There are wizards that enable you to:

- ▶ Create Virtual Host
- ▶ Install New Application
- ▶ Create JDBC Provider
- ▶ Create Data Source
- ▶ Deploy IBM Telephone Directory
- ▶ Deploy IBM Survey Creator

Under the wizards, you see links to configure some of the server properties, which include:

- ▶ Server Tracing
- ▶ Server Ports
- ▶ View HTTP Servers (associated with a virtual host for this application server)

In addition, you can manage:

- ▶ Installed Applications
- ▶ Virtual Hosts
- ▶ JDBC Providers
- ▶ Data Sources

Under Problem Determination in the navigation pane, you may view logs. And under Tools, you may launch the Administrative Console from this pane.

3.3.3 Advanced tab

The Advanced tab (Figure 3-17) contains advanced tasks that you can perform on your servers. It contains the four subtabs Settings, Internet Users and Groups, Search Setup, and TCM. The Advanced tab allows you to:

- ▶ Set the Global Server Settings, which are values that apply to all HTTP Server (powered by Apache) configurations
- ▶ Work with Internet users and groups to create, delete, and populate validation lists
- ▶ Use the Webserver Search Engine page to set up your Web site for full text searches on HTML and text files
- ▶ Set up your iSeries server with advanced cache management servers called the *Triggered Cache Manager (TCM)*

Settings subtab

In addition to the Common Tasks and Wizards links described in “Setup tab: Common tasks and wizards” on page 37), the Settings subtab contains the Global Server Settings (Figure 3-18). The values for Global Server Settings apply to each IBM HTTP Server (powered by Apache) configuration. The values provided here can be overridden individually within each IBM HTTP Server (powered by Apache) configuration.

The screenshot shows the 'Global Server Settings' page in the IBM Web Administration for iSeries interface. The top navigation bar includes 'Setup', 'Manage', 'Advanced' (selected), and 'Related Links'. Below this, the 'Settings' subtab is active, with other subtabs being 'Internet Users and Groups', 'Search Setup', and 'TCM'. On the left, a tree view shows 'Common Tasks and Wizards' expanded, with 'Global Settings' and 'Global Server Settings' listed. The main content area is titled 'Global Server Settings' and contains several configuration fields: 'Autostart' (set to 'No'), 'Number of threads' (Maximum: 40), 'Coded character set identifier' (00819), 'Server mapping tables' (Outgoing EBCDIC/ASCII table: *CCSID, Incoming ASCII/EBCDIC table: *CCSID), and 'Library' fields. At the bottom, there are 'OK', 'Apply', and 'Cancel' buttons.

Figure 3-18 IBM Web Administration for iSeries: Global Server Settings

Note: Since these values have the potential to impact the overall usability and performance of all your servers, you must understand the implications of making any changes to the global settings.

To change the global server configuration parameters, you click Global Server Settings link in the left navigation pane as shown in Figure 3-18. You can also use the Change HTTP Attributes (CHGHTTPA) CL command as shown in Figure 3-19.

Change HTTP Attributes (CHGHTTPA)		
Type choices, press Enter.		
Autostart	*NO	*YES, *NO, *SAME
Number of server threads:		
Minimum	10	1-9999, *SAME, *DFT
Maximum	80	1-9999, *SAME, *DFT, *NOMAX
Coded character set identifier	00819	1-65533, *SAME, *DFT
Server mapping tables:		
Outgoing EBCDIC/ASCII table .	*CCSID	Name, *SAME, *CCSID, *DFT
Library		Name, *LIBL, *CURLIB
Incoming ASCII/EBCDIC table .	*CCSID	Name, *SAME, *CCSID, *DFT
Library		Name, *LIBL, *CURLIB

Figure 3-19 Change HTTP Attributes display

Internet Users and Groups subtab

The Internet Users and Groups subtab allows you to define users of the HTTP servers using *validation lists*. You may also list or delete digital certificates. You can perform the following tasks on this subtab:

- ▶ Add Internet User
- ▶ Change Internet User Password
- ▶ Delete Internet User
- ▶ List Internet Users
- ▶ Delete Certificate
- ▶ List Certificates

Validation lists are used in conjunction with other resources to limit access to server resources. Each validation list contains a list of Internet users and passwords. You use the Internet users and groups form to list and manage digital certificates associated with validation lists.

A validation list is used to store user ID and password information about remote users. You can use existing validation lists or create your own. Validation list entries also require you to identify an authentication protocol type to associate with the user ID and password. Validation lists are case sensitive and reside in iSeries libraries.

A *group file* identifies a group of users with a common security profile. It contains iSeries user profiles. A *user profile* is an object with a unique name that contains the user's password, the list of special authorities assigned to a user, and the objects the user owns or has access to.

Note: Validation lists are not the same as iSeries user profiles. User profiles must be created manually and are independent of a validation list.

A *digital certificate* is a form of personal identification that can be verified electronically. Only the certificate owner who holds the corresponding private key can present a certificate for authentication through a Web browser session. The key can be validated through any readily available public key. You use the Digital Certificate Manager to create, distribute, and manage digital certificates.

For more information about authentication and other security topics, see Chapter 6, “Defending the IFS” on page 101.

Add Internet User

You can use the Add Internet User form (Figure 3-20) to add user names and passwords for server access by using validation lists or group files or both. Internet users exist independently of OS/400 user profiles and are used only with the IBM HTTP Server.

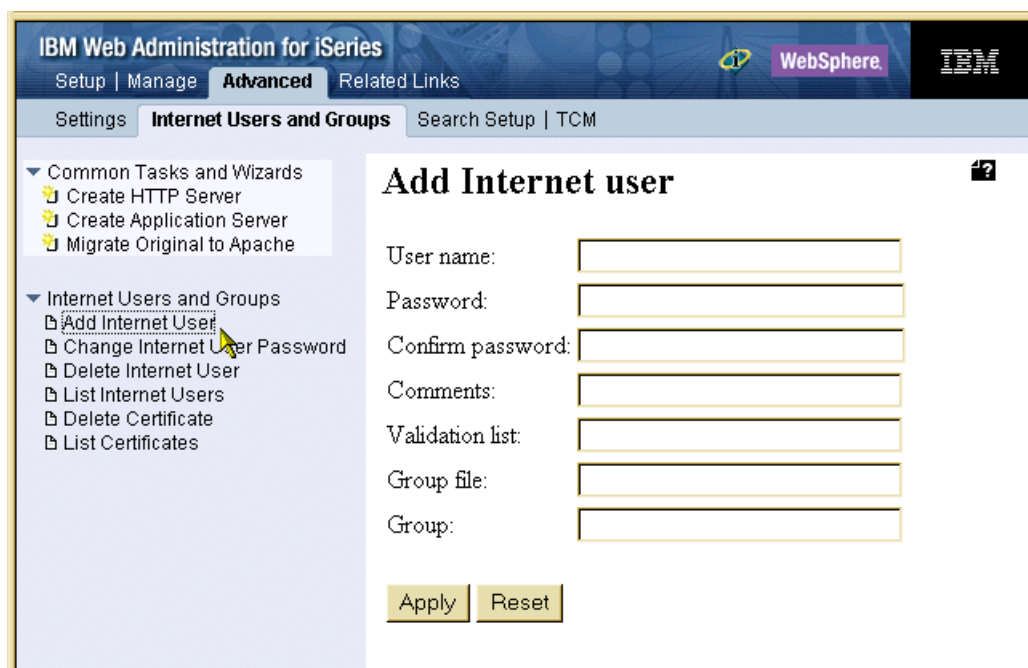
The screenshot shows the 'Add Internet user' form within the IBM Web Administration for iSeries interface. The top navigation bar includes 'Setup | Manage | Advanced | Related Links'. The left sidebar has a tree view with 'Common Tasks and Wizards' and 'Internet Users and Groups'. The 'Add Internet User' option is selected. The main content area contains the following fields: 'User name:', 'Password:', 'Confirm password:', 'Comments:', 'Validation list:', 'Group file:', and 'Group:'. Each field has a corresponding text input box. At the bottom of the form are 'Apply' and 'Reset' buttons. The IBM logo and 'WebSphere' branding are visible in the top right corner.

Figure 3-20 IBM Web Administration for iSeries: Add Internet User

To enter a new user, you perform the following steps:

1. Enter the new user name that you want to add. The user name can be up to 10 characters long. You can specify names that contain a blank or certain special characters in them. The special characters that you can use include tab, colon (:), comma (,), parenthesis (), at sign (@), exclamation point (!), and close brace (}). For example, you could specify the name “Smith, Joe”.
2. Enter the password for the new user. The password can be up to 10 characters long. This is an optional entry.
3. In the Confirm password field, enter the same password as in the previous step for verification. The confirmed password can be up to 10 characters long and must match exactly your entry in the Password field. This is an optional entry.
4. Enter any comments to provide additional information about the user. For example, you can enter the user’s e-mail address. This is an optional entry.

5. Enter a validation list for which to add the user. A validation list is an OS/400 object of type *VLDL that stores user names and passwords for use in access control. Validation lists are case sensitive and reside in OS/400 libraries. They are required when you add a user, unless you are adding the user to a group file. If you enter a validation list that does not exist, the system will create it for you.

Note: Enter the validation list name in the format somelib/somelist. In this example, *somelib* is an existing library in the QSYS file system (up to 10 characters long). *somelist* is the name of a validation list (up to 10 characters long) that exists in that library.

6. Enter an existing group file directory with a fully qualified path name in the IFS, followed by the group file name (in the format /somedir/group1.grp). A *group file* contains information about which users belong to which groups. A group file is created if it does not already exist. This entry is required only if you are adding a user to a group file instead of a validation list. Group file names can be case sensitive, depending on which file system they are located in, and cannot contain any blank characters.

Note: Group files are supported only in the IFS.

7. Enter the group within the group file in which to add the user. A *group* is a collection of users who require common access control to a directory or file. This may, for example, be a collection of people in the same department. If you enter a group that does not already exist in the group file, then it is created for you. A group must be specified when adding a user to a group file. A group cannot contain blank characters.

Note: When adding an Internet user to a group file, if a group is specified, then the user is added directly into that group.

8. Click **Apply** to update the configuration file with the information that you entered on the form. Click **Reset** to return to the values that were on the form before you made the changes.

Note: This form does not use any configuration directives.

Change Internet User Password

You can use this form to change the password for an existing Internet user. Internet users exist independently of OS/400 user profiles. They are used only with IBM HTTP Server when you perform the following steps:

1. Enter the user name for which to change the password. The user name can be up to 10 characters long.
2. Enter the new password for the user. The password can be up to 10 characters long.
3. In the Confirm password field, enter the same password as above for verification. The confirmed password can be up to 10 characters long and must match exactly your entry in the New Password field.
4. Enter an existing validation list that stores the user name and password that you want to change. As mentioned earlier, a validation list is an OS/400 object of type *VLDL that stores user names and passwords for use in access control. Such lists are case-sensitive and reside in OS/400 libraries. They are required when you change the password of a user and check for valid user names and passwords.

5. Click **Apply** to update the configuration file with the information that you entered on the form. Click **Reset** to return to the values that were on the form before you made the changes.

Delete Internet User

You can use the Delete Internet User form to delete an Internet user from a validation list, group file, group, all groups within a group file, or all of these at the same time. Internet users exist independently of OS/400 user profiles and are used only with IBM HTTP Server.

List Internet Users

You can use the List Internet Users form to list the Internet users in a specific validation list. Internet users exist independently of OS/400 user profiles and are used only with IBM HTTP Server.

Delete Certificate

You can use the Delete Certificate form to delete certificates associated with a validation list. Validation lists are used in conjunction with protection setups and access control lists to limit access to your server resources. They contain a list of users and their passwords. You use this form to delete certificates that users are authorized to on a particular validation list.

Digital certificates handle authentication together with digital signatures. *Authentication* is the process used to verify the identity of an Internet user. Digital signatures and certificates provide integrity and accountability for Internet users.

List Certificates

You can use the List Certificates form to list any certificates associated with a validation list. Certificates within validation lists can be used in conjunction with protection setups.

Search Setup subtab

The Search Setup subtab contains a series of forms that allow you set up the *IBM Webserver Search Engine*. Use the Webserver Search Engine to set up your Web site for full text searches on HTML and text files.

The following forms are available to help you setup and manage the IBM Webserver Search Engine:

- ▶ Create search index
- ▶ Update search index
- ▶ Merge search index
- ▶ Delete search index
- ▶ View status of search index
- ▶ Build document list
- ▶ Register document list
- ▶ Delete document list
- ▶ Work with document list status
- ▶ Build URL mapping rules file
- ▶ Build thesaurus dictionary
- ▶ Test thesaurus dictionary
- ▶ Retrieve thesaurus definition
- ▶ Delete thesaurus dictionary
- ▶ Build URL object
- ▶ Delete URL object
- ▶ Build options object
- ▶ Delete options object

- ▶ Build validation list
- ▶ Delete validation list
- ▶ Search index

You can control which options are available to the user and how the search results are displayed with an included Net.Data macro. The Net.Data macro also allows you to customize the look and feel of your Web site. There is a sample Net.Data macro and a sample HTML file in the /QIBM/ProdData/HTTP/Public/HTTPSVR directory to use and test.

You can enhance your search results when you create the index by using field support for META tags and a mapping rules file to map internal file names to an external path. The search forms contain additional ways to customize a search, such as using a thesaurus.

Before you can search your files, you must have an index. The *index* is a set of files that contain the contents of the documents (in a searchable form) that are to be searched.

The *search index* is used by the search engine rather than searching all of the actual documents. It is created based upon a document list. A *document list* contains a list of fully qualified path names of all the documents that you want to index. You can create the document list from files in a local directory by entering a path name or from files on another server by using the Web crawling functions. The Web crawling functions allow you to search a single URL or a list of URLs, including those requiring authentication.

Before you add a search engine to your Web site, you can test it here and can see forms that are similar to the ones you can customize for your web site.

For more information about the IBM Webserver Search Engine, see Chapter 11, “Getting started with Webserver Search Engine and Web Crawler” on page 307.

TCM subtab

The TCM subtab (Figure 3-21) allows you to create and configure a TCM server that can work with your HTTP Server (powered by Apache) to dramatically improve the response time for complex Web pages. These pages allow you to set up your iSeries system with advanced cache management servers called *Triggered Cache Manager servers*.

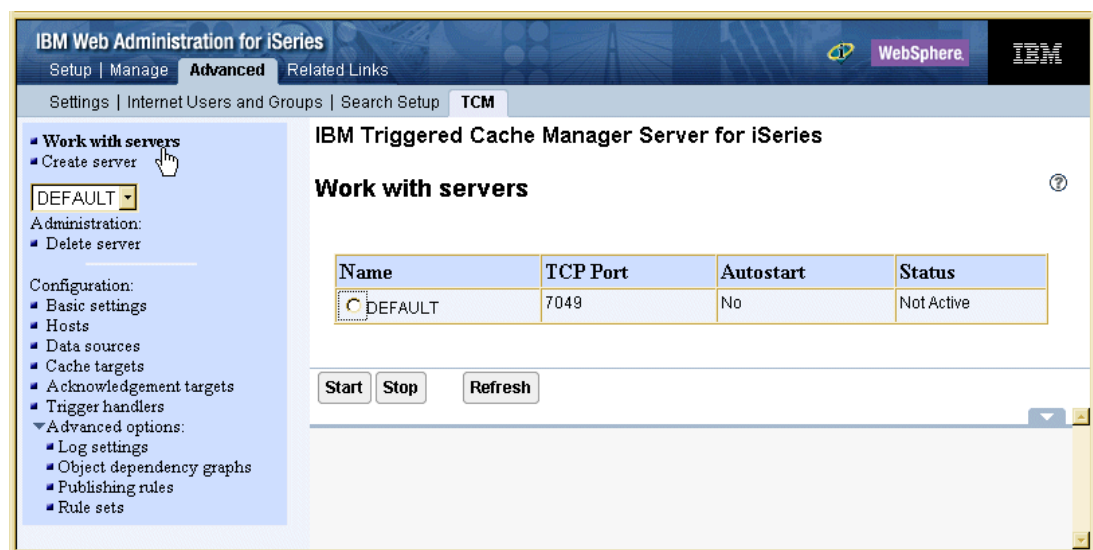


Figure 3-21 IBM Web Administration for iSeries: TCM tab Work with servers panel

You can use Triggered Cache Manager servers in conjunction with Web servers and Web document caching agents to keep Internet (and intranet) Web sites running at peak performance. If your Web site contains dynamically produced Web pages, or perhaps rapidly changing static pages, you may want to have a Web document caching agent that is managed by a Triggered Cache Manager server.

You use the menu of links in the navigation frame on the left to locate and load configuration and administration forms for Triggered Cache Manager servers. To work with a specific server, you select its name from the selection list located toward the top of the menu. You can load forms for a specific server by clicking links that appear under the selection box after you select a name.

An empty selection box means that there are currently no servers to work with. In this case, you must create one using the Create server link above the selection box.

Clicking links in the navigation frame loads forms in this frame, called the *work area frame*. Some links are grouped under a single menu heading. Clicking such a heading expands the group of links for display. Clicking the same heading again collapses the group and hides the links.

You can find more information about TCM in 10.5, “Triggered Cache Manager” on page 259.

3.3.4 Related links page

On the Related Links page (Figure 3-22), you simply follow the links listed on the page for more information about the HTTP Server for iSeries and other related products.

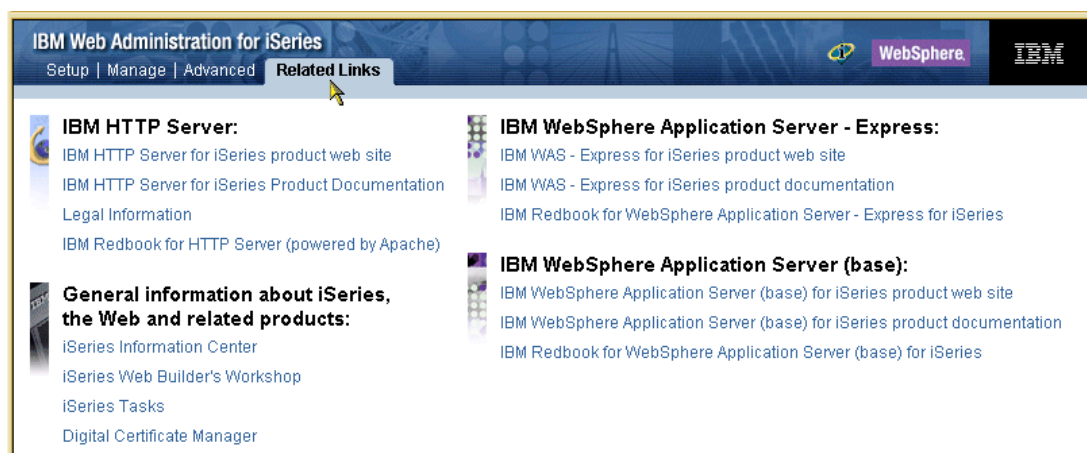


Figure 3-22 IBM Web Administration for iSeries: Related Links page



Quick guide to Apache contexts and request routing

Let us step back a bit from the HTTP Server (powered by Apache) implementation on the iSeries to learn more about generic version 2.0 Apache configuration directives and request routing.

The best place to learn about Apache configuration directives is from the many books written by the world's Apache experts. That is one of the advantages of using the world's most popular Web servers. We recommend that you read the following books as a start to your Apache library (in addition to this IBM Redbook, of course!):

- ▶ *Professional Apache 2.0* by Peter Wainwright, with Michael Link and Poornachandra Sarang
- ▶ *Apache Server 2 Bible with CD-ROM* by Mohammed J. Kabir
- ▶ *Apache Server 2.0 The Complete Reference* by Ryan Bloom, with a forward by Brian Behlendorf

This chapter introduces you to the basic concepts of configuration and request routing with your HTTP Server (powered by Apache). Remember, HTTP servers are essentially file servers. The configuration directives tell the server which files to serve and which to protect. Then you see how to apply what you learned by stepping through a simple configuration scenario using the administration graphical user interface (GUI).

4.1 In-context configuration

If you are familiar with the HTTP Server (original), you will find configuring the HTTP Server (powered by Apache) a little different. Apache server configurations generally deal directly with files in physical directories. This differs from the HTTP Server (original) method, which relies on Uniform Resource Locator (URL) mapping and deals only indirectly with physical file locations. URL mapping lets you hide the physical location of Web objects, which is a security advantage. However, in the Apache world, the thinking is that a simpler approach to configuration reduces errors that may otherwise compromise security.

Because it's unlikely that you would want to protect all the files on your iSeries server in the same manner, the Apache server provides a mechanism for subsetting the configuration file into logical entities. Apache's configuration subsets are called *contexts*. You can think of a context as a container for settings.

For example, the *configuration structure* (see Figure 4-2 on page 65 for a graphical representation of this) of the server ITSOCO may look like this:

```
ITS099 global configuration
Directory /
Directory /itso/itso99/itsoco
Directory /itso/itso99/itsoco/downloads
```

The entire ITSOCO configuration is itself considered a context, the *global context*. Apache subdivides that context into sub-contexts in the same way subdirectories divide the root directory in a hierarchical file system. In fact, the global context is bound to the document root directory of the ITSOCO server. Apache defines all contexts in relation to the document root.

The contexts that you will use most often are the *directory* and *file contexts*. Directory contexts define configuration settings for an entire directory. File contexts define settings for files matching a particular name pattern.

In the ITSOCO configuration structure, the first context within the global context is of type directory. This context defines settings for the document root directory and all its subdirectories. Similarly, the context directory /itso/itso99/itsoco is a directory context that defines settings for the /itso/itso99/itsoco document directory and all its subdirectories. The context directory /itso/itso99/itsoco/downloads further defines or overrides settings.

Within directory contexts, file contexts let you define or override settings based on file names or extensions. File contexts have the form files *pattern*, where *pattern* is an expression that matches file names by name and extension (for example, *.gif).

Other context types serve special purposes, so you may use them less frequently. The *location* context specifies the URL of a request to further refine or override settings. Another powerful context is *VirtualHost*. In the same way that the directory context defines how the Apache server treats a group of files (and files located in all subdirectories), the VirtualHost context defines settings based on the Internet Protocol (IP) address and port that a client uses to access your iSeries server. When the server receives a request for a document on a particular virtual host (defined by IP address and port), the VirtualHost context (for example, VirtualHost 1.2.30.40:port) supplies the configuration directives used by the server.

Directives within a directory context let users retrieve the contents of files in the directory exactly as those files appear on disk. VirtualHost contexts let you change this behavior to force file retrieval through a specific data filter, such as encryption. You can define a VirtualHost context with a specific IP address and port (for example, the default Secure Sockets Layer (SSL) and Transport Layer Security (TLS) port 443) to force the file to transfer via an SSL/TLS encryption session.

4.2 Apache server request routing

The IBM HTTP Server (powered by Apache) filters incoming URL requests by applying certain access rules and configuration values specified in the server configuration. This process is called *request routing*.

The HTTP Server (powered by Apache) can map a URL to a directory or file with the `mod_alias` directives, which are processed sequentially. These directives include `Alias(Match)`, `ScriptAlias(Match)`, `Redirect(Match | Temp | Permanent)`, and `Rewrite`. `ScriptAlias` lets you map to programs that reside outside of `DocumentRoot`. For example, you can use `ScriptAlias` to map `/cgi-bin/` to `/QSYS.LIB/yourlib.lib/db2www.pgm`. By default, Apache configurations are not case sensitive.

However, most Apache request processing uses a procedure called a *directory walk*. In this procedure, the server reads contexts in a specific order and merges the settings of specified by the directives in those contexts. In the ITSOco configuration structure, for example, the server first reads all directives within the global context, then those within directory `/` (the root directory), then those within `/itso/itso99/itsoco`, and then those within `/itso/itso99/itsoco/downloads`. All the while, it is merging the settings of the directives that it finds. This is a more powerful mechanism than the HTTP Server (original) `Pass/Exec` syntax. It lets you organize directives hierarchically (the same way you organize Web content) and apply directives more consistently to groups of similar files.

The directory walk merges directives from contexts in the following order:

Tip: The directory context (number 1) is the weakest and is more likely to be overridden by stronger contexts such as the files (number 3) and sections inside of the `VirtualHost` context (number 5).

1. Directory contexts (except those containing regular expressions) and `.htaccess` files are merged simultaneously (with `.htaccess` files overriding directory). Regular expressions are a UNIX shorthand method of expressing ranges of objects. You can think of these as an advanced version of DOS pattern-matching characters.

.htaccess file: This is an optional local configuration file that is described on the following page.

2. `DirectoryMatch` and directory contexts that contain regular expressions are merged.
3. Files and `FilesMatch` contexts are merged simultaneously.
4. Location and `LocationMatch` contexts are merged simultaneously.
5. Sections (that is, nested directory, files, location, and limit contexts) inside `VirtualHost` contexts are applied merged after the corresponding sections outside the virtual host definition. This lets virtual hosts override the main server configuration.

In the merging process, lower-level contexts that occur later in the sequence can inherit or replace settings from earlier higher-level contexts. Or they can override those settings. Directives that apply to subdirectories can override those for parent directories.

Each directory can have its own local configuration file that you specify with the `AccessFileName` directive. The normal convention is to use `.htaccess` as the file name, but for security reasons some webmasters change it to something less well known. The file name should start with a period, which makes it a hidden file. An `.htaccess` file can override the server configuration only for the contents of the directory in which the file resides. Using `.htaccess` files complicates the configuration and security. You can learn more about this in 4.4, “Configuration recommendations” on page 63.

4.3 Request routing example

Let’s look at an example directory. If you expand the configuration contexts for the server `ITSOco`, you may see the following directives:

```
<Directory />
    AllowOverride None
    order deny,allow
    deny from all
</Directory>
<Directory /itso/itso99/itsoco/downloads>
    order deny,allow
    allow from 10.10.0.0/255.255.0.0
    deny from all
    AlwaysDirectoryIndex On
    DirectoryIndex index.html
    Options +Indexes
</Directory>
<Directory /itso/itso99/itsoco>
    order allow,deny
    allow from all
</Directory>
```

Let’s say a client request for the URL `/itso/itso99/itsoco/downloads/downloads.html` arrives from IP address `10.10.1.2`. The first directory match, because it’s the shortest, is “/” (the root directory). The directive `AllowOverride None` tells the server not to look for the `.htaccess` file in this directory (or any subdirectory) unless there’s a specific override. This directive improves performance and sets an important security precedent.

The `order` directive defines the order in which Apache evaluates the list of clients to which you deny or allow access. (No top-down processing here.) Specifically, `order deny,allow` means that the default allows access, but this is overridden by any `deny` directives, which in turn, can be overridden by any `allow` directives. In this case, `deny from all` overrides the default `allow` access, and since no specific `allow` directive is used, `deny from all` is the rule for this directory (and all subdirectories unless specifically overridden). At the top of this configuration hierarchy, no access is allowed unless we override at a lower level. This approach secures the server by default.

Next, the request routing process examines the directory `/itso/itso99/itsoco`. We want to allow open access to `DocumentRoot`, which contains our `ITSOco` server’s home page. To do this, we override and reverse the directive that we gave in the previous directory (`order deny,allow`) with the directive `order allow,deny`. This directive establishes `deny` as the default, but lets the next directive (`allow from all`) override denial of access.

The final context (because it is the longest) is directory `/itso/itso99/itsoco/downloads`. Here again we override the settings for the order directive by issuing the directive `order deny,allow`. This directive sets `allow` as the default, but the next directive `deny from all`, which excludes all clients, overrides it. The last directive to apply is `allow from 10.10.0.0/255.255.0.0`. (We ignore the final three directives in the directory here.) This directive allows access only to those clients with IP addresses within the 10.10.0.0 subnet and denies all others, which receive error message 403 "Forbidden by Rule".

This directory walk from the shortest to the longest is a useful tool that lets you set security precedents that you can override when necessary and create relatively compact, yet powerful configurations.

Note: Although we use a directory walk to demonstrate how the HTTP Server (powered by Apache) handles requests, this is not done at runtime for each request of the server. The HTTP Server (powered by Apache) reads the configuration file with all its directives at startup time once and builds a tree structure to hold all the configuration details.

4.4 Configuration recommendations

Nested contexts and configuration directives can be confusing. Here are a few rules to help you keep track of how the server processes them:

- ▶ Don't use Location sections unless you really must do so. There are times when they are unavoidable, for example, when you're configuring servlets with Tomcat. However, you can use Directory sections for almost everything you need to do.
- ▶ Use File sections only when you really need them. You can solve most problems by putting files into a separate directory and using a Directory section.
- ▶ Minimize the number of sections in the configuration file. Rearranging the directory structure can help you accomplish this.

As we mentioned earlier, you can use a special file with the default name `.htaccess` to override settings in a specific directory context. However, overuse of this file can impair performance and widen security holes. You should limit your use of `.htaccess` files to those situations in which you need distributed administration and configuration. Avoid using several `.htaccess` files in the same directory path (for example, `/www/.htaccess` and `/www/html/.htaccess`).

4.5 Configuring directory listings

Let's examine some powerful concepts of the HTTP Server (powered by Apache) configuration GUI. When you request the URL `http://system_name:8099` (where `system_name` is the name of your system), you configure the server to display the home page, usually a file named `index.html` in the directory defined as `DocumentRoot /ITSO/ITSO99/ITSOco`. We override this default behavior to display the contents of a subdirectory (that is, `/ITSO/ITSO99/ITSOco/Downloads`) instead of the Web page. Apache displays the contents as a list of files and attributes with file names automatically made "hot" so that clicking one downloads the corresponding file.

On the configuration page (Figure 4-1), in the Server area list on the right, you select the context you want to work with and then select the forms and wizards you need in the menu of options in the left hand navigation area. We don't have a context for the `/ITSO/ITSO99/ITSOco/Downloads` subdirectory, so we must create one.

Follow these steps (note that the step numbers correspond to those in Figure 4-1):

1. From the Server list on the left, select **your server name**. In this example, we select **ITS099**. From the Server area list on the right, select **Global Configuration**.
2. In the left pane, under Server Properties, select **Container Management**.
3. On the right side, select the **Directories** tab.
4. Under the directory table, click **Add**. In the list in the Type column, select **Directory**. Enter a new directory. For our example, we used `/its0/its099/itsoco/downloads`.
5. Click **Continue**.
6. Click **OK**.

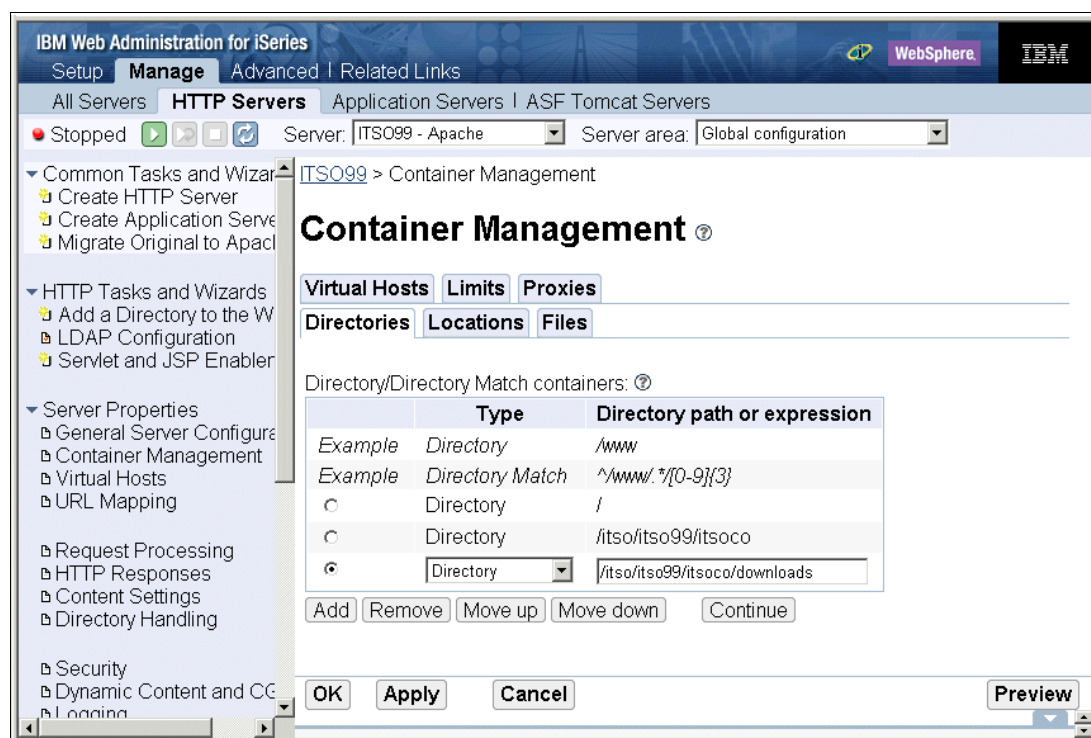


Figure 4-1 Configuring directory listings: Directories page

The Server area list below the tabs at the top of the page is immediately updated with the new context directory `/itso/itso99/itsoco/downloads`. It should look like the example in Figure 4-2.

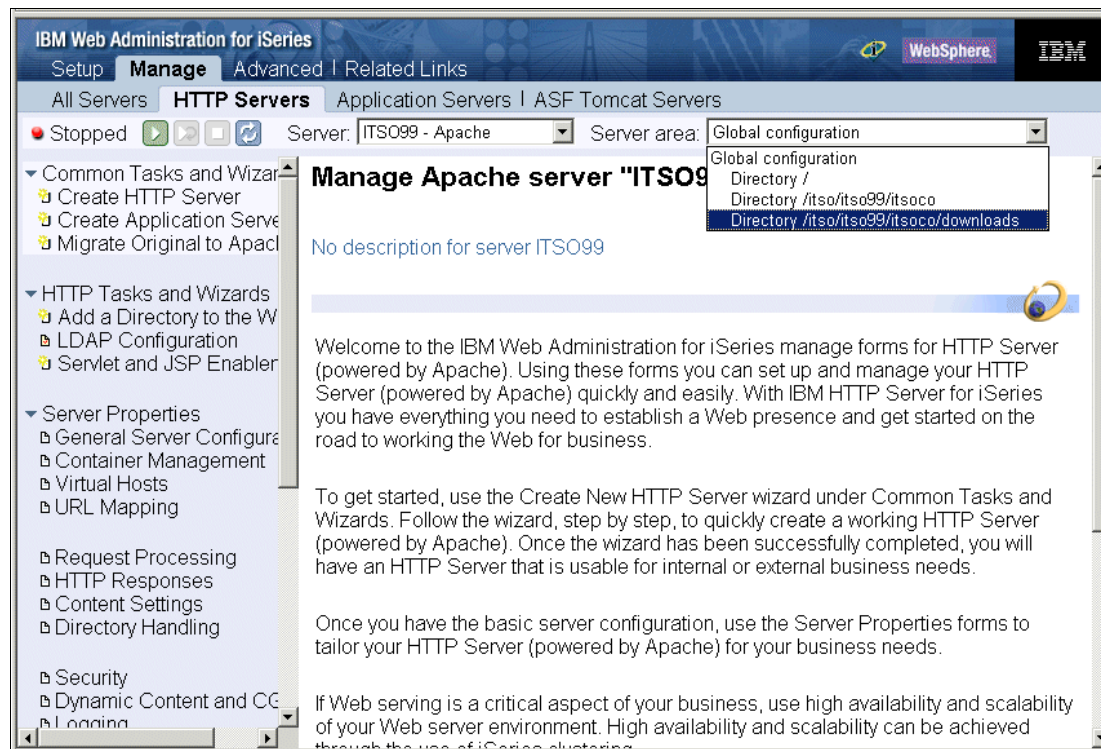


Figure 4-2 Configuring directory listings: Success at creating a new directory context

Next, you enable directory listings in the subdirectory Downloads as shown in Figure 4-3. For security reasons, you don't want your server to display directory listings by default. Follow these steps (note that the numbers correspond to those in Figure 4-3):

1. From the Server area list, select **Directory /its0/its099/itsoco/downloads**.

Notice that the groups of available configuration tabs changed immediately. If you didn't see the change, change your selection to **Global configuration**, and repeat the procedure. This is an important feature of the configuration GUI. That is, when you select an Apache configuration context (directory, files, VirtualHost and so on), the user interface shows only the tabs that you can apply to that context. Tabs that are not available are grayed out.

2. From the left pane, under Server Properties, select **Directory Handling**.
3. You see a new set of tabs on the right. Select the **General Settings** tab.
4. Select **Display directory listing for all directories**.
5. Click **OK**.

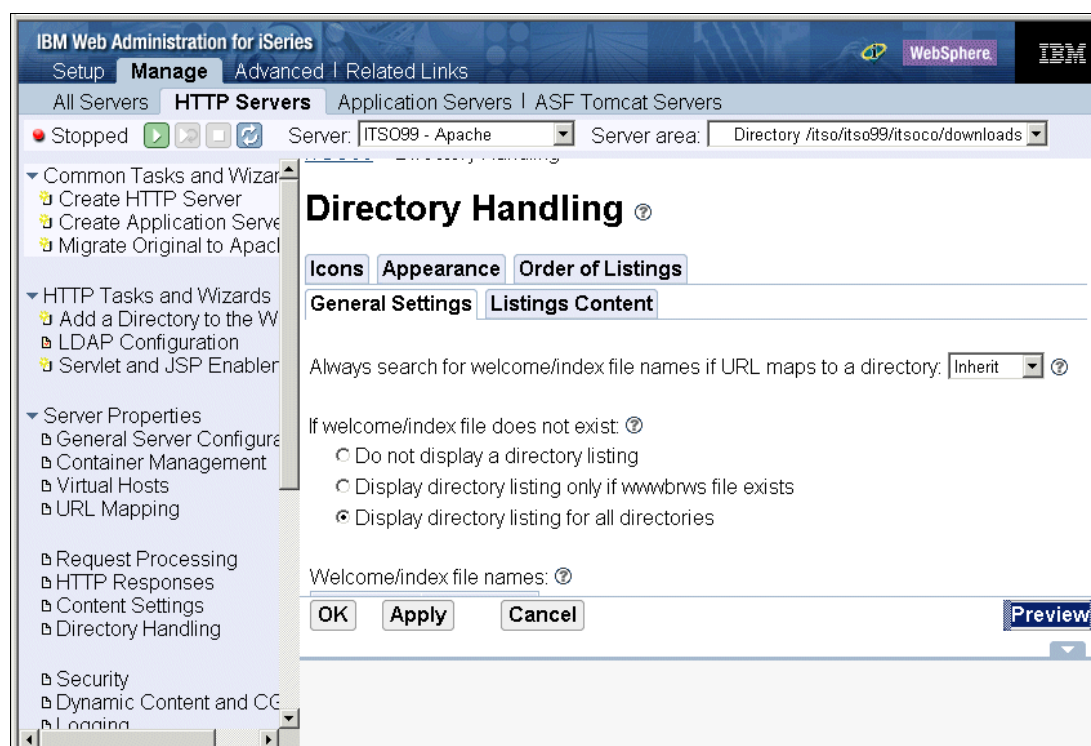


Figure 4-3 Enabling Directory Listings

As shown in Figure 4-4, you can review your changes by selecting from the left pane **Tools** → **Display Configuration File**. We recommend that you do this every time you make a change to your configuration.

Tip: Another nice feature of the HTTP Server (powered by Apache) is the Preview button located in the lower right corner of the GUI. Anytime you make a configuration change or entry, you may click this button and it displays the configuration prior to saving it.

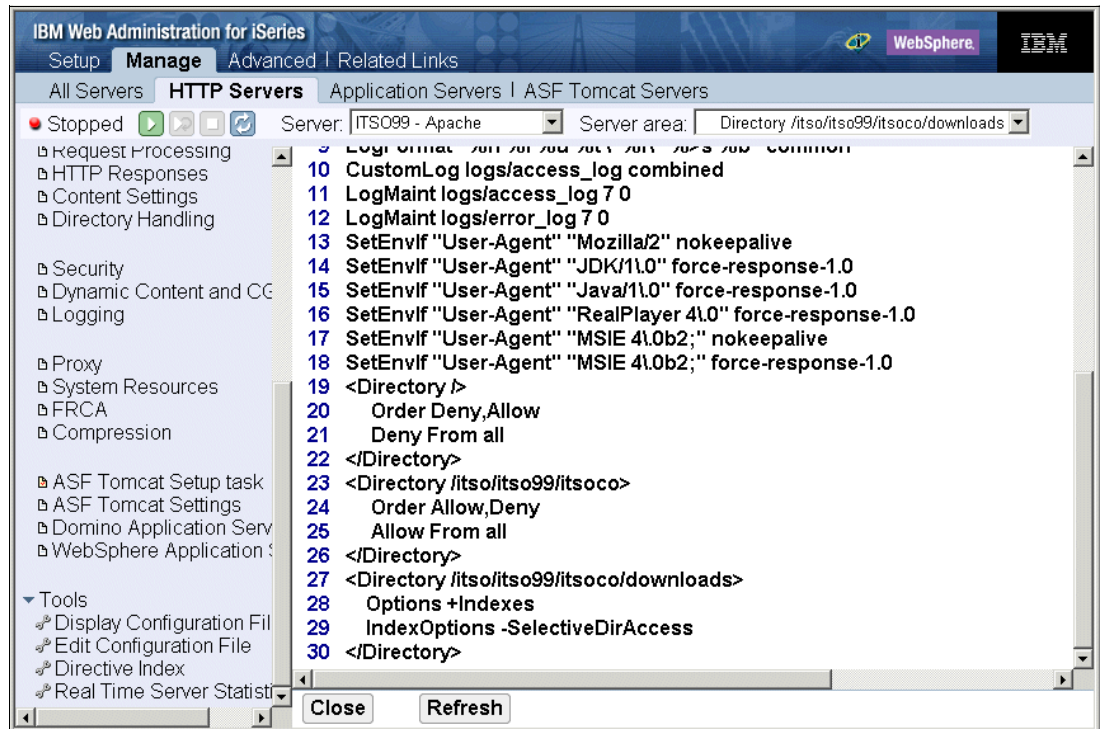


Figure 4-4 The new configuration file

You should now test the new configuration:

1. Restart the server.
2. Click **Refresh** to ensure that the server stays started. Sometimes errors in the configuration can cause the server to stop.
3. Enter the following URL to test your directory listing:

`http://as20:8099/downloads/`

You see a listing of files in the Downloads subdirectory much like the example in Figure 4-5. For extra practice, you can make the directory listings “fancy” by using the *Fancy/Customized Indexing* form in the Web Site Definition form group.

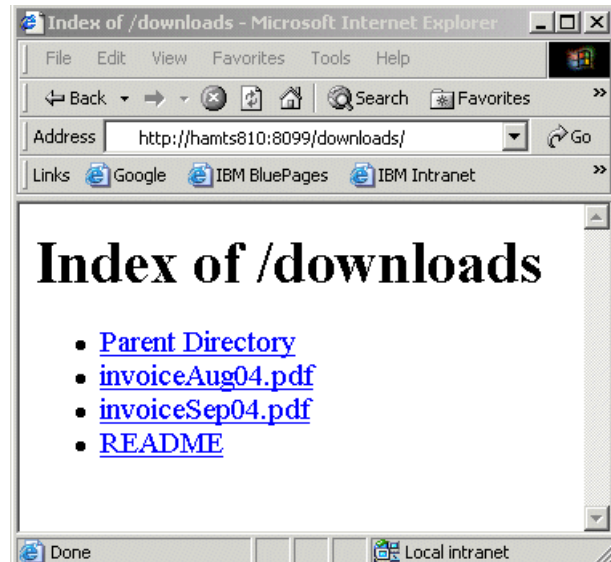


Figure 4-5 Configuring directory listings: List of hot links to files in /downloads



Part 2

How to...

As you prepare to use HTTP Server (powered by Apache), you may ask these questions:

- ▶ What is the best way to configure my Apache Web server to support multiple virtual hosts, and how do I do it?
- ▶ How do I implement security with HTTP Server (powered by Apache)?
- ▶ If I want to do more than just serve static Web pages, how do I serve the dynamic data?

This part answers these questions and more. It instructs you on how to use virtual hosts, secure your server, and serve dynamic data with server-side includes (SSI), Net.Data, and Common Gateway Interface (CGI).

Virtual hosts

The concept of *virtual hosts* in terms of Web serving refers to the practice of maintaining more than one domain in a single server. The way the domains are primarily differentiated is by their host name or Internet Protocol (IP) address. Therefore, client requests are routed to the correct domain by IP address or by host name contained in the Uniform Resource Locator (URL) header. Traditionally virtual host implementation requires as many HTTP servers running simultaneously as domains that the system is going to serve. Figure 5-1 illustrates the virtual host concept.

One of the most important features of the HTTP Server (powered by Apache) is the way this concept is implemented. The HTTP Server (powered by Apache) allows you to use *one* HTTP server to host as many domains as the environment requires.

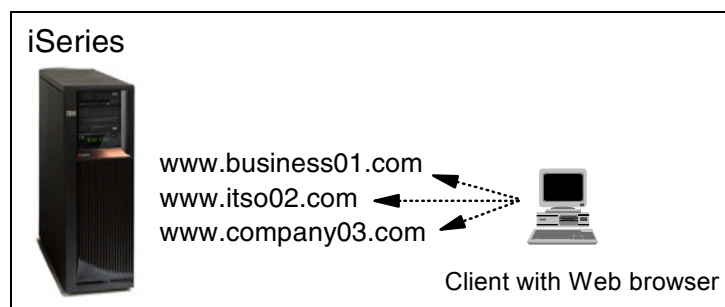


Figure 5-1 Virtual host concept

The virtual host concept is primarily used by Internet Server Provider (ISPs), content providers, or companies that need to manage multiples domains, but they do not want to use a different server for each domain they want to serve. For example, if two companies want to establish presence on the Internet without buying, building, and maintaining their own Web site, they can ask an ISP to host and publish their Web pages. The ISP then sets up the virtual host implementation so that each site looks like it is running on a different server. Each of these servers is called a *virtual host* since they are really running on the same server.

Before implementing the virtual host concept using the HTTP Server (powered by Apache), let us review some of the concepts introduced by the HTTP Server (original) in this area and some information required for you to understand the HTTP Server (powered by Apache) implementation.

5.1 HTTP virtual host overview

If you need to use the iSeries server to host multiple domains, you need to think about:

- ▶ The way Transmission Control Protocol/Internet Protocol (TCP/IP) is configured
- ▶ The way the HTTP server will be configured
- ▶ The way the HTTP server will handle visitor requests

5.1.1 The way TCP/IP is configured

Your TCP/IP configuration depends on the number of physical network connections your server has, for example:

- ▶ One single IP address over one physical connection
- ▶ Multiple IP addresses over one physical connection
- ▶ Multiple IP addresses over multiple physical connections

Your iSeries server can be configured as a multi-homed server with multiple IP interfaces or virtual IP addresses. Any of those IP addresses can be used with the HTTP server to handle multiple domains in one iSeries server. For an alternative look at how to set up any of those TCP/IP approaches, see the IBM Redbooks *Application Service Provider Business Model: Implementation on the iSeries Server*, SG24-6053, and *iSeries IP Networks: Dynamic!*, SG24-6718.

For your virtual host configuration, you need to identify the IP address, port, and domain name for each domain.

5.1.2 The way the HTTP server will be configured

The HTTP server can be configured to host:

- ▶ Multiple domains using one HTTP server
- ▶ Multiple domains using multiple HTTP servers, one for each domain
- ▶ A combination of both

Figure 5-2 shows two approaches you can use with your HTTP server to support multiple domains.

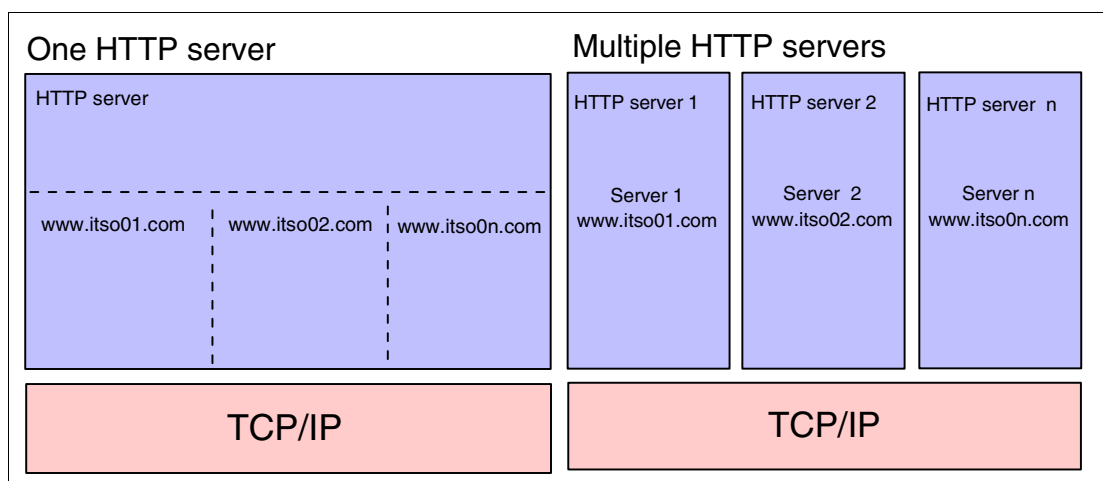


Figure 5-2 HTTP server instance approaches

The HTTP configuration is different if the domains are served using one HTTP server or multiple HTTP servers. Each approach requires a different process and is based on different server directives, as explained in the Table 5-1.

Table 5-1 One and multiple server instances: Configuration comparison

One HTTP server	Multiple HTTP servers
One configuration file shared by all domains	Multiple configuration files, one for each domain
One or many access log and error log files	Each configuration file includes its access and error log files
One or many document root directories	Many document root directories, one per configuration file
One or multiple IP addresses and ports within the server instance	One or multiple IP addresses and ports per configuration file
The HTTP server process uses the IP address or the domain to serve the request	The HTTP server process uses the IP address or the domain name to serve the request
Specific HTTP server directives to handle each domain	No specific server directives since each domain has its own configuration file

The runtime environment of each approach also differs, as demonstrated in Table 5-2.

Table 5-2 One and multiple server instances: Running environment comparison

One HTTP server	Multiple HTTP servers
All domains run under the same environment: configuration file, process, and some service directives.	Each domain runs under its own environment: configuration file, process, and server directives.
One process is started because there is only one HTTP server.	Many processes are started since more than one HTTP server exists.
All the domains run under the same user profile, since there is only one server process. However, each one can be configured under a different security mechanism. Refer to Chapter 6, “Defending the IFS” on page 101, for more information.	Each domain can run under its own user profile, since there is one process per domain. You can change the user profile on which the server instance runs, using the ServerUserID directive.
If you have problems with one domain, and the recovery procedure implies the HTTP server restart, all the domains served by the HTTP are restarted.	If you have problems with one domain, the HTTP server associated with the domain can be restarted without affecting any other domain.

The information in Table 5-1 and Table 5-2 allows you to identify the differences between the two approaches. According to these tables, both HTTP server approaches process visitor requests correctly. Deciding between running under one or multiple HTTP servers usually depends more on the system resources, such memory and Central Processing Unit (CPU). It also depends on security issues, such as independency between domains. HTTP server directive limitations that can be used with any approach are not as important.

Unless, the multiple HTTP server approach allows you to achieve any specific requirement, we recommend that you use *one* HTTP server to host the domains, since you must only create and maintain one configuration file. The iSeries server has only one HTTP server to process the request, which saves memory and CPU resources for other system activities.

5.1.3 The way the HTTP server will handle visitor requests

The client submits a request. Based on information found in the header, the HTTP server will process the request. The HTTP server can use the domain name provided in the header or the translated IP address, depending on:

- ▶ The number of IP addresses your system has
- ▶ The way your IP addresses are used by the system

The iSeries server can host intranet and Internet domains. If the iSeries server is used as a Web server in the Internet, you have to register every domain the system will use, which means more cost associated with the HTTP implementation.

- ▶ The version of the HTTP protocol supported in the environment considering HTTP Version 1.0 and HTTP Version 1.1

The HTTP Server (powered by Apache) and most of the Web client browsers support the HTTP 1.1 protocol. The differences between the protocols are:

- With the HTTP 1.0 protocol, the HTTP server relies on the Domain Name System (DNS) server to translate the domain name into the IP address. The HTTP server then uses the IP address to process the visitor request.
- With the HTTP 1.1 protocol, the domain name is included in the visitor request (as a header). Therefore, the HTTP server receives the domain name and can process the request according to the HTTP directives to include into the configuration file.

Because not all the client browsers support the HTTP Version 1.1 protocol, the HTTP server must be configured in a way that, regardless of the limitations the environment has, each client request is handled correctly. To accomplish this, the HTTP Server (powered by Apache) supports three different virtual host implementations:

- ▶ **IP based:** The HTTP server uses the IP address to handle a visitor's request. It can be used with HTTP 1.0. For more information about the IP-based implementation, see 5.3, "Virtual hosts: IP-based implementation" on page 77.
- ▶ **Name based:** The HTTP server includes the domain name into the URL header to handle visitor requests. This requires HTTP 1.1. To learn more about name-based implementation, see 5.4, "Virtual hosts: Name-based implementation" on page 89.
- ▶ **Mass dynamic based:** The HTTP server retrieves the domain name provided in the URL header to process the data requested by the client. This is done dynamically, which means that the domain does not have to be registered using any <VirtualHost> context in the HTTP configuration. To learn more about mass dynamic-based implementation, see 5.5, "Virtual hosts: Mass dynamic implementation" on page 94.

Based on the TCP/IP configuration and the Web client browser capabilities, the configuration options listed in Table 5-3 are available to create the virtual host implementation.

Table 5-3 Instance creation options

IP address and port	Protocol	HTTP server configuration approach
One IP address, one port	HTTP 1.0	▶ One HTTP server, with no virtual host configuration, can be created
One IP address, one port	HTTP 1.1	▶ One HTTP server, without virtual host configuration ▶ One HTTP server, name based ▶ One HTTP server, mass dynamic
Any other IP address and port combination	HTTP 1.0 HTTP 1.1	▶ One HTTP server, IP-based ▶ Multiple HTTP servers ▶ One HTTP server, mass dynamic

With this information about TCP/IP configuration options, you can design your own virtual host implementation to serve your domains. The “ideal” virtual host implementation using the HTTP Server (powered by Apache) includes:

- ▶ One HTTP server, using specific virtual host directives in the configuration file
- ▶ Using either the IP address or the domain name to serve visitor requests
- ▶ Using any of the TCP/IP configuration approaches

Table 5-4 summarizes the HTTP virtual host implementation options. The HTTP Server (original) column is included only as an aid.

Table 5-4 Virtual host implementation

Server type	HTTP Server (original)	HTTP Server (powered by Apache)
IP interfaces	One or multiple IP interfaces	One or multiple IP interfaces
Server implementation	One or multiple instances	One or multiple HTTP server
Server directives	Pass	Listen <VirtualHost> NamedVirtualHost VirtualDocumentRoot VirtualScriptAlias
Server variations	IP base Name based	IP-based Name based Mass dynamic

Note: The <VirtualHost> implementation is supported since Apache Version 1.1, but it was rewritten by Apache Version 1.3. The HTTP Server (powered by Apache) supports Apache Version 2.0.

Although the iSeries server supports the HTTP Server (original) and HTTP Server (powered by Apache), the HTTP Server (original) configuration is beyond the scope of this book. If you want to learn more about the HTTP Server (original), concepts and configuration process, refer to the HTTP Server iSeries Information Center at:

<http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.htm?info/rzaie/rzaiemain.htm>

Now, let’s focus on the HTTP Server (powered by Apache) implementation and how it works.

Note: Whether you use one server instance or multiple server instances, if any of the domains require the Secure Sockets Layer (SSL)/Transport Layer Security (TLS) encryption, you have to use specific virtual host directives within the configuration files. That’s because as the SSL configuration needs to listen on a different port, usually 443, for encrypted information. For additional information of how to create the SSL/TLS virtual host directive, see 6.4, “Encrypting your data with SSL and TLS” on page 127.

5.2 HTTP Server (powered by Apache) virtual host overview

The virtual host implementation with the HTTP Server (powered by Apache) allows one HTTP server to process the request for one or more domains. Configuring your HTTP server for a virtual host is a two-stage process:

1. Assign the IP address and port to listen on.
2. Define and configure the virtual host configuration.

The virtual host configuration is done using the `<VirtualHost>` context directive. `<VirtualHost>` and `</VirtualHost>` are used to enclose directives that apply only to a particular virtual host. Almost any configuration directive can be used within the `VirtualHost` context, with the exception of directives that control process creation. A typical virtual host configuration looks similar to the following example:

```
Listen 10.5.92.28:8002
<VirtualHost 10.5.92.28:8002>
DocumentRoot /itso/itso01/ITS0co
ServerName www.itso01.com
</VirtualHost>
```

Here *10.5.92.28* is the IP address, *8002* is the port number the virtual host listens on, and *www.itso01.com* is the domain name.

Tip: The `<VirtualHost>` context defines a specific IP address and port through which clients access files on your iSeries server. Through the design of your TCP/IP network (complete with routers, firewalls, physically separated public and private networks, and so on), you can force certain clients to arrive on a specific IP address or port. An example of this is intranet traffic on 10.1.1.1 and Internet traffic on 100.1.1.1. In this way, you can treat these clients differently by adding specific directives within the `<VirtualHost>` context.

In terms of the HTTP Server (powered by Apache) configuration, there are two different approaches to set up the virtual host:

- ▶ **Main server with `<VirtualHost>`:** One or more domains are handled by the main server configuration and the `<VirtualHost>` context for any specific domain requirements, such as security.
- ▶ **Only `<VirtualHost>`:** Use the `<VirtualHost>` context for domain and leave the main server with no requests to handle.

With the first approach, unless overridden by `<VirtualHost>` context, the main server behavior is inherited by all the virtual hosts. For example, the `ServerAdmin` and `ErrorLog` directives are used by all virtual hosts.

```
Listen 10.5.92.28:8002
ServerAdmin admin@company.com
ErrorLog /itso/logs/error_log
NameVirtualHost 10.5.92.28:8002

<VirtualHost 10.5.92.28:8002>
DocumentRoot /www/itso01/itsoco
ServerName www.itso01.com
</VirtualHost>

<VirtualHost 10.5.92.28:8002>
DocumentRoot /itso/itso02/itsoco
ServerName www.itso02.com
</VirtualHost>
```

With the second approach, each virtual host has its own server directive. Therefore, there is no main server configuration that applies to all virtual hosts.

```
Listen 10.5.92.28:8002
ServerAdmin admin@company1.com
NameVirtualHost 10.5.92.28:8002

<VirtualHost 10.5.92.28:8002>
DocumentRoot /itso/its01/itsoco
```

```

ServerName www.itso01.com
ErrorLog /itso/itso01/logs/error_log
</VirtualHost>

```

```

<VirtualHost 10.5.92.28:8002>
DocumentRoot /www/itso02
ServerName www.itso02.com
ErrorLog /itso/itso02/logs/error_log
</VirtualHost>

```

Now when a request arrives, the HTTP server uses the IP address and port, or the domain name it arrived on, to find a matching virtual host configuration. If no virtual host matches the address and port (or the domain name), the request is handled by the main server configuration. If it does match a virtual host directive, the HTTP server uses the configuration of that <VirtualHost> to handle the request.

5.2.1 Additional resources

Here are some additional references that can help guide your study of virtual hosts as implemented by the HTTP Server (powered by Apache):

- ▶ Apache Week Web site
<http://www.apacheweek.com/features/vhost>
- ▶ Apache virtual host documentation
<http://httpd.apache.org/docs/vhosts/index.html>

5.3 Virtual hosts: IP-based implementation

As the term IP-based indicates, the IP virtual host implementation is based on the way the HTTP server uses the IP address to serve the domain. If you want to serve multiple domains using this implementation, the server must have a different IP address or port for each IP-based virtual host.

You can do this by either having multiple physical network connections or having multiple IP addresses. Figure 5-3 shows the multiple IP addresses and multiple physical connections concept. TCP/IP implementation on the iSeries server supports the implementation of multiple IP addresses.

For additional information, refer to *TCP/IP Configuration and Reference*, SC41-5420, to create an additional IP interface in one physical connection.

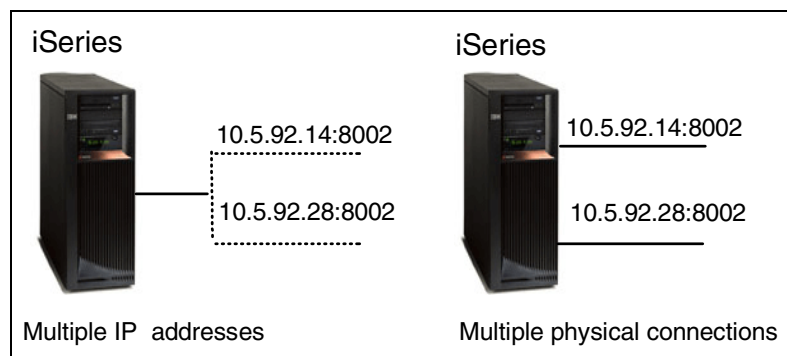


Figure 5-3 Multiple IP addresses and physical connections concept

After you identify the IP address for each domain, you tell the HTTP Server (powered by Apache) how to handle it using the `<VirtualHost>` directive. IP-based implementation works very well but requires a dedicated IP address for every virtual host the system is going to serve. Usually this means more cost especially if you must purchase these additional IP addresses and domains from an ISP. The IP-based virtual host implementation provides an immediate solution for any browser since the implementation does not rely on any specific browser functionality. Therefore, it tends to be the preferred method for many sites to implement virtual hosting. From the browser point of view, there is no difference between a virtual host and a real host. Both have their own server name and associated IP address, as shown in Figure 5-4.

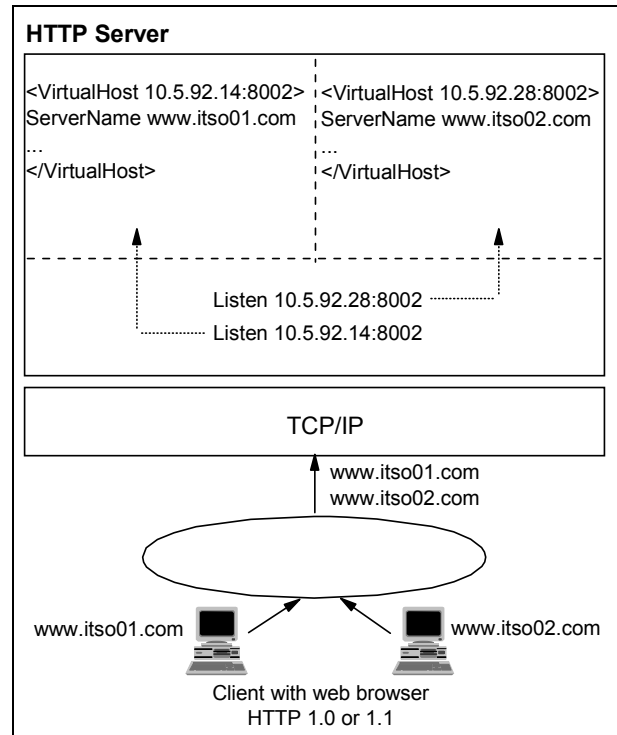


Figure 5-4 IP-based implementation

IP-based implementation supports multiple domains using different IP addresses. This is a good implementation approach if you want to run each domain in a different network using its own IP address.

5.3.1 IP-based virtual host: Problem scenario

Your company needs to host two different domains, `www.itso01.com` and `www.itso02.com` using one iSeries server. Since your iSeries server has two available IP addresses and the Web browser clients do not all support the HTTP 1.1 protocol, you decide to create an IP-based virtual host implementation.

Figure 5-5 shows the problem scenario.

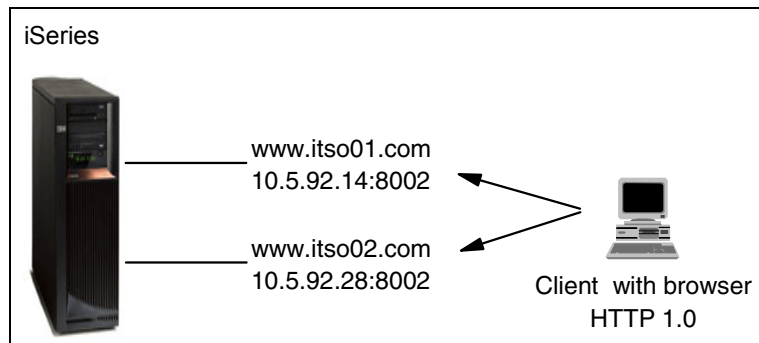


Figure 5-5 IP-based virtual host: Problem overview

To configure the HTTP Server (powered by Apache) to handle the client requests, you must identify some of the basic resources used by the HTTP server to route and serve those domains. Table 5-5 outlines some of the basic resources used by the HTTP server.

Table 5-5 Basic Web server resources

Resource	ITSO01	ITSO02
ServerName	www.itso01.com	www.itso02.com
Welcome page	index.html	index.html
IP address	10.5.92.14:8002	10.5.92.28:8002
DocumentRoot	/itso/itso01/itsoco	/itso/itso02/itsoco
ErrorLog	/itso/itso01/logs/error_log	/itso/itso02/logs/error_log

With the Web server resources identified and the IP addresses selected, it is time to create the HTTP configuration that will route each visitor request to the appropriate Web site.

5.3.2 IP-based virtual host: Solution overview

We already know what an IP-based virtual host can do. Now it is time to see the global configuration steps involved in the process to create, activate, and use the IP-based virtual host. Setting up basic TCP/IP operations is beyond the scope of this book, since we only cover the additional steps required to set up the IP-based virtual host. You need to configure the HTTP Server (powered by Apache) to route `www.itso01.com` and `www.itso02.com` requests to the appropriated IP-based virtual host. Four steps are involved in the IP-based virtual host implementation as shown in Figure 5-6 (note that the step numbers correspond to those in Figure 5-6):

1. Add the IP addresses and ports to which your HTTP server will listen. Also add the server host name.
2. Under the General Server Configuration, add a document root.
3. Create a new directory from which the files will be served.
4. Create an error log file.

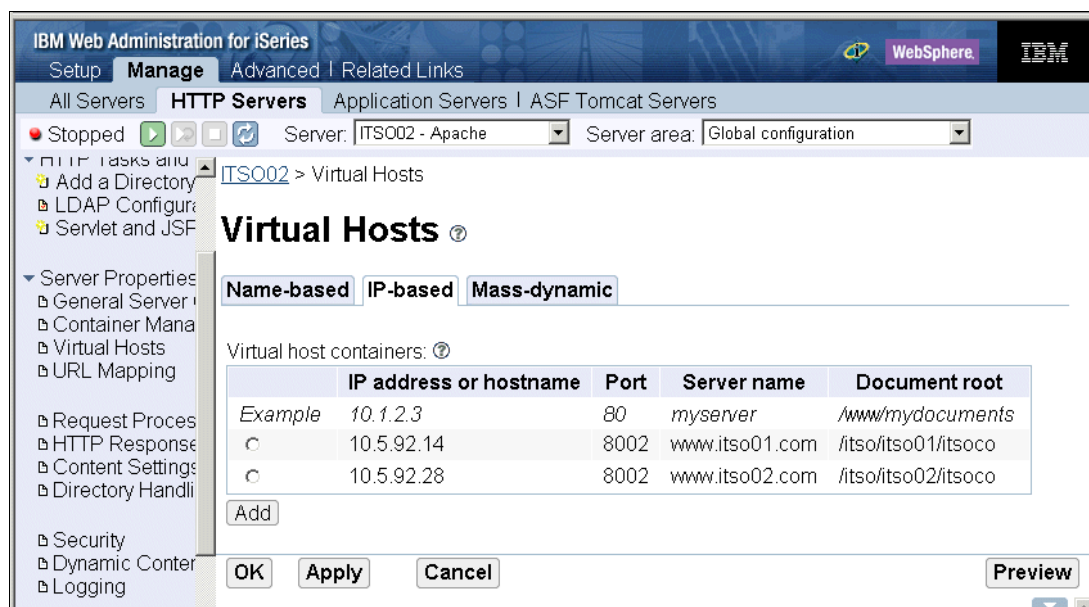


Figure 5-6 IP-based configuration steps

Table 5-6 offers a detailed guide to the steps used to create a sample IP-based configuration on your iSeries server. It shows how to create your first IP-based virtual host implementation. It includes a column for the HTTP Server (original) for a comparison of configuration directives. The procedure to create additional virtual host contexts works the same way.

Table 5-6 IP-based configuration overview

Original configuration	Apache GUI configuration steps	Apache final configuration file
<ul style="list-style-type: none"> ► DNS entries to resolve IP addresses. For example: www.itso01.com 10.5.92.14:8002 ► Pass directives: pass /itso01 10.5.92.14:8002 pass /itso02 10.5.92.28:8002 	<p>Add Listen for any new IP addresses (2 and 3)</p> <p>Create a virtual host context:</p> <ul style="list-style-type: none"> ► One for IP address 10.5.92.14:8002 (24) ► Second for IP address 10.5.92.28:8002 (38) <p>Populate virtual host contexts:</p> <ul style="list-style-type: none"> ► Add the DocumentRoot for 10.5.92.14:8002 (25). ► Add the ServerName for 10.5.92.14:8002 (26). ► Add the ErrorLog for 10.5.92.14:8002 (29, 30). ► Add the directory (31, 36). ► Add the DocumentRoot for 10.5.92.28:8002 (39). ► Add the ServerName for 10.5.92.28:8002 (40). ► Add the ErrorLog for 10.5.92.28:8002 (43, 44). ► Add the directory (45, 50). 	<pre> 1 # Configuration originally created by 2 Listen 10.5.92.28:8002 3 Listen 10.5.92.14:8002 ... 24 <VirtualHost 10.5.92.14:8002> 25 DocumentRoot /itso/itso01/itsoco 26 ServerName www.itso01.com 27 UseCanonicalName Off 28 HostNameLookups off 29 ErrorLog /itso/itso01/logs/error_log 30 LogLevel error 31 <Directory /itso/itso01/itsoco> 32 AllowOverride None 33 order allow,deny 34 allow from all 35 </Directory> 36 Alias /itso01/ /itso/itso01/itsoco/ 37 </VirtualHost> 38 <VirtualHost 10.5.92.28:8002> 39 DocumentRoot /itso/itso02/itsoco 40 ServerName www.itso02.com 41 UseCanonicalName Off 42 HostNameLookups off 43 ErrorLog /itso/itso02/logs/error_log 44 LogLevel error 45 <Directory /itso/itso02/itsoco> 46 AllowOverride None 47 order allow,deny 48 allow from all 49 </Directory> 50 Alias /itso02/ /itso/itso02/itsoco/ 51 </VirtualHost> 52 ... </pre>

5.3.3 IP-based virtual host: Step-by-step implementation

To create the <VirtualHost> context and the server directives for each domain, follow these steps (note that the step numbers correspond to those in the figures that follow):

1. Start the administration GUI.
2. Select the **Manage** tab.
3. From the Server list, select the HTTP server you want to work with. For this example, we select **ITSO02**. In the Server area list, select **Global configuration**.
4. To set the IP addresses and ports on which the server will listen, click **General Server Configuration** on the left navigation pane.

5. In the General Server Configuration panel (Figure 5-7), click the **General Settings** tab.

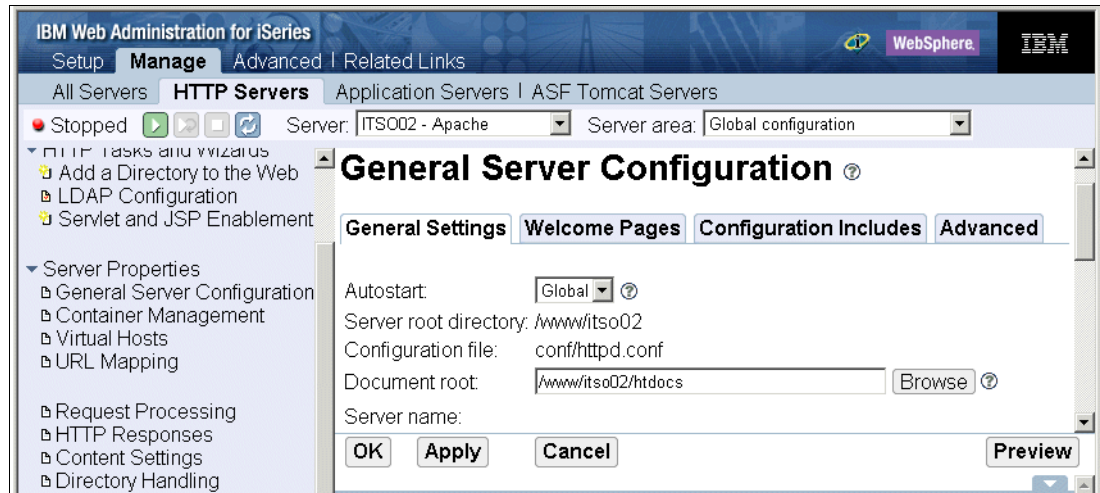


Figure 5-7 IP-based virtual hosting: Adding listen directives

- a. Scroll down until you see the “Server IP address and ports to listen on” heading. Click **Add**.
- b. Enter the IP address and port on which your server will listen. In this example, we type 10.5.92.28 for the IP address and 8002 for the port. Depending of the values you enter in the basic server instance creation, there may already be some values under IP address and port to listen on. Then click **Continue**.
- c. Repeat steps a and b for any additional IP address and ports. Then click **OK**.

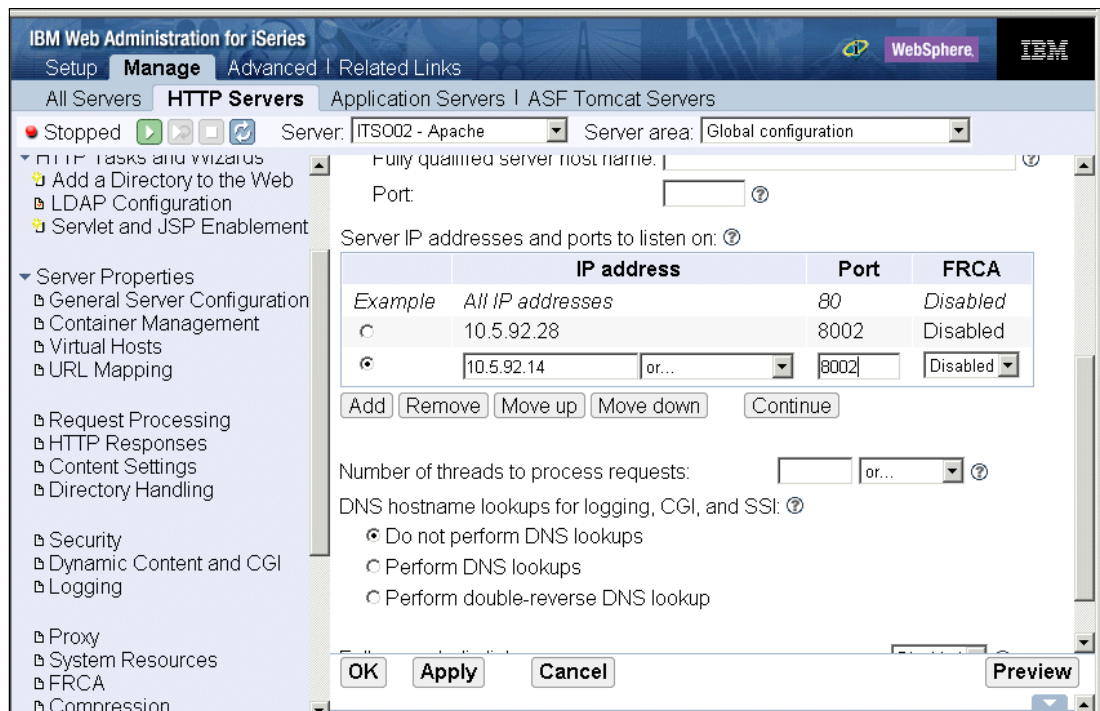


Figure 5-8 IP-based virtual hosting: Port listen directives

When you finish, the HTTP configuration file has some new entries, one for each IP address and the port that you added, as shown in Figure 5-9.

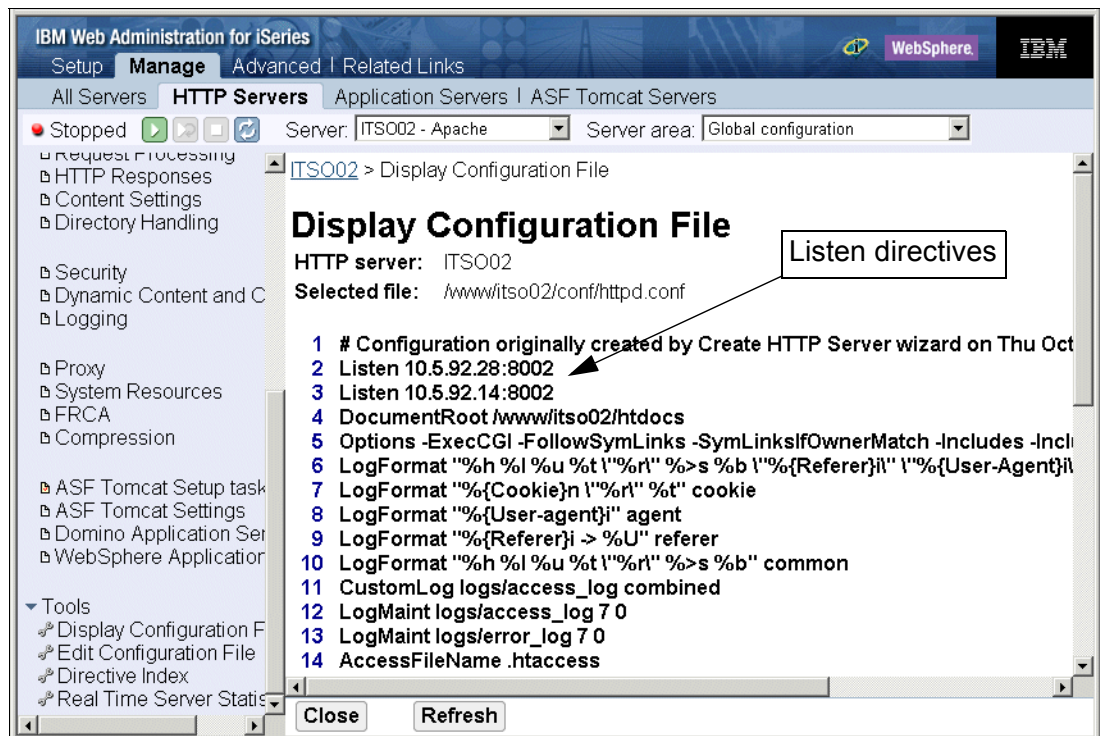


Figure 5-9 IP-based virtual hosting: Listen directives entries

6. Create the IP-based virtual host context. Here you create the context where all the directives related to a specific domain will be located.
 - a. In the left pane, under Server Properties, click **Virtual Hosts**.
 - b. In the Virtual Hosts panel (Figure 5-10), select the **IP-based** tab.
 - c. Click **Add**.
 - d. Enter the IP address and port. For this example, we enter 10.5.92.14:8002 for the first domain we are going to serve. Enter the server name and the document root for the new virtual host. When a request comes in for the specified IP address and port, the index and other HTML documents are served from this document root directory. For the first virtual host, we entered /itso/itso01/itsoco.
 - e. Click **Continue**.

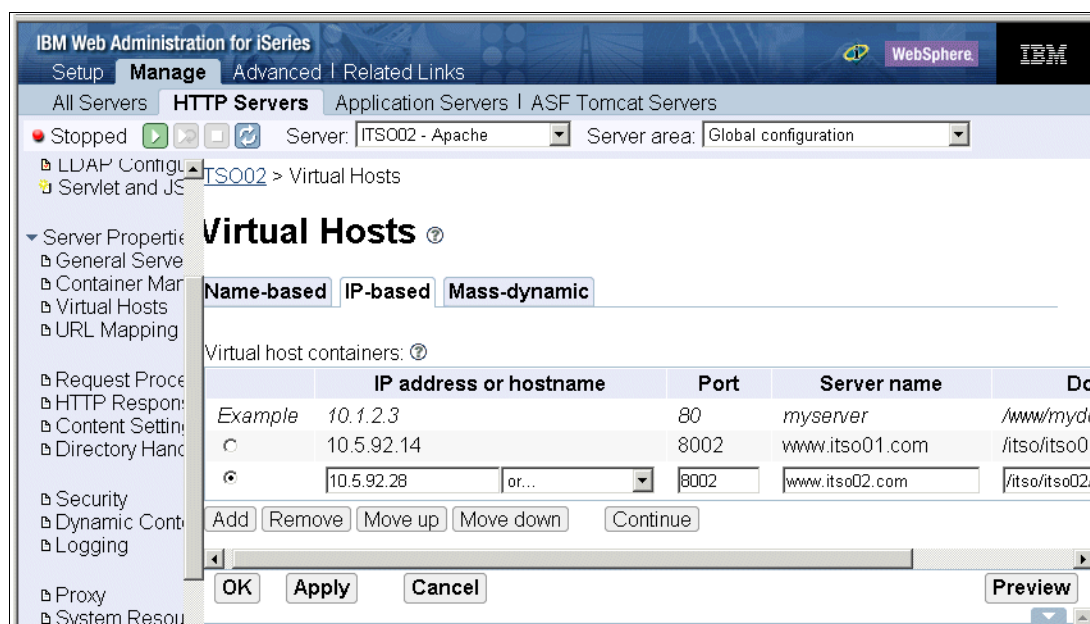


Figure 5-10 IP-based virtual hosting: General configuration

- f. Repeat steps c, d, and e for each IP-based virtual host context that you want to create.

- g. When you are done, click **Apply** or **OK**. Both the administrative GUI and the HTTP configuration file have new virtual host entries (one for each IP virtual host you created). Figure 5-11 shows the new virtual hosts entries.

Note: Clicking Apply saves the changes and remains on the form. Clicking OK also saves the changes but exits the form.

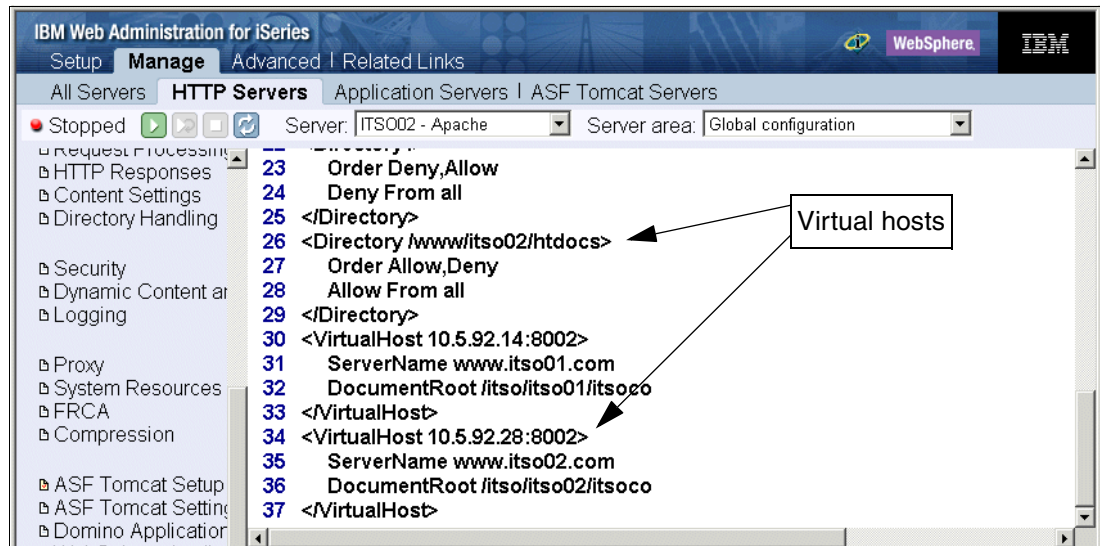


Figure 5-11 IP-based virtual hosting: Virtual hosts entries

7. Populate the IP virtual host context. To populate the `<VirtualHost>` context, add the necessary directives within the `<VirtualHost>` context. Here you add those directives as specified in Table 5-6 on page 80 for each virtual host.

The administrative GUI has added the DocumentRoot, ServerName, and some default settings under each `<VirtualHost>` context you configured. You can add or change as many HTTP server directives as you want under the `<VirtualHost>` context.

Tip: There are some directives that you cannot use in a virtual host. To determine whether a particular directive can be used in the `<VirtualHost>` context, click the **Manage** tab. Under Tools in the left pane, click **Directive Index**.

8. Create a new directory from which the files will be served. Perform the following steps to add a directory to the virtual host context.
 - a. From the Server area list, select the virtual host context you want to work with. For this example, we select **Virtual Host 10.5.92.14:8002**.
 - b. In the left pane, under Tasks and Wizards, select **Add a Directory to the Web** as shown in Figure 5-12.

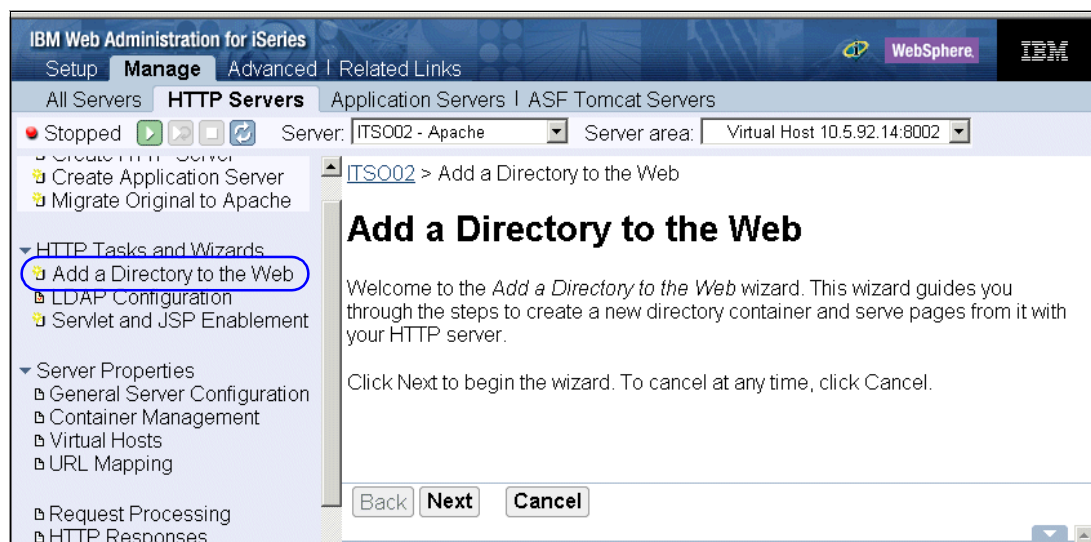


Figure 5-12 IP-based virtual hosting: Add a directory to the Web

- c. Follow the wizard to create the new directory entries. This wizard asks for the information related to the directory where the files are located, as shown in Table 5-7.

Table 5-7 Serving new directory options

Wizard question	Option or value
What type of information do you want to serve from this directory?	<ul style="list-style-type: none"> ► Static Web pages and files ► Common Gateway Interface (CGI)
Which directory do you want to serve from?	Enter the directory name, for example /itso/itso01/itsoco
What alias do you want to use to access the files in this directory?	Enter any alias, for example /itso01/

If the directory you include does not exist, the wizard creates it for you. If the directory exists, the wizard uses it. Then you can copy your HTML code, images, etc. into this directory.

After you follow the wizard and enter the information from the table, you should see the page shown in Figure 5-13.

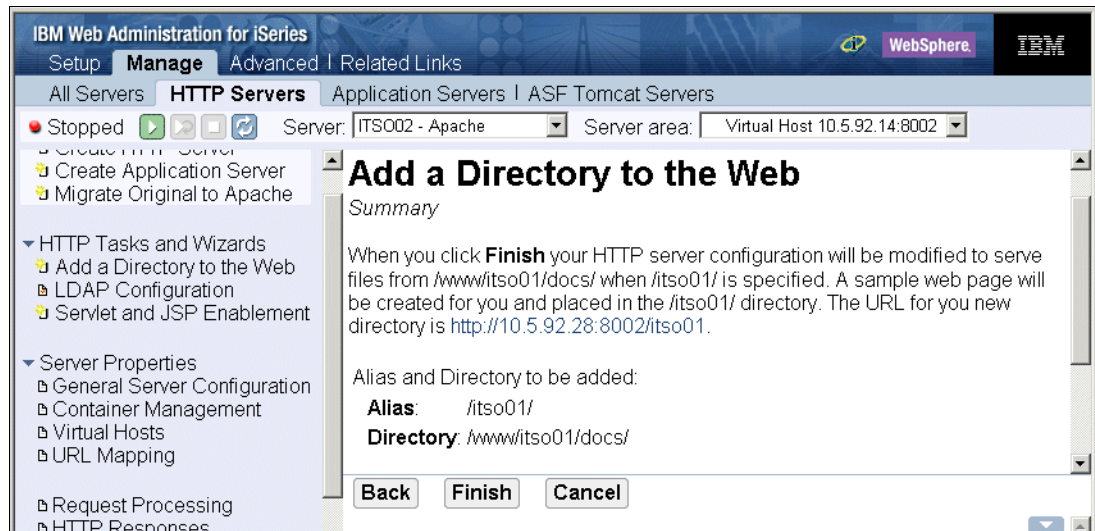


Figure 5-13 IP-based virtual hosting: Adding a directory to the Web

d. Click **Finish**.

The administrative GUI and HTTP configuration file now have new entries for the new directories, as shown in Figure 5-14. In addition to the Directory entry, there is an Alias entry, since the directory can be a server using an alias.

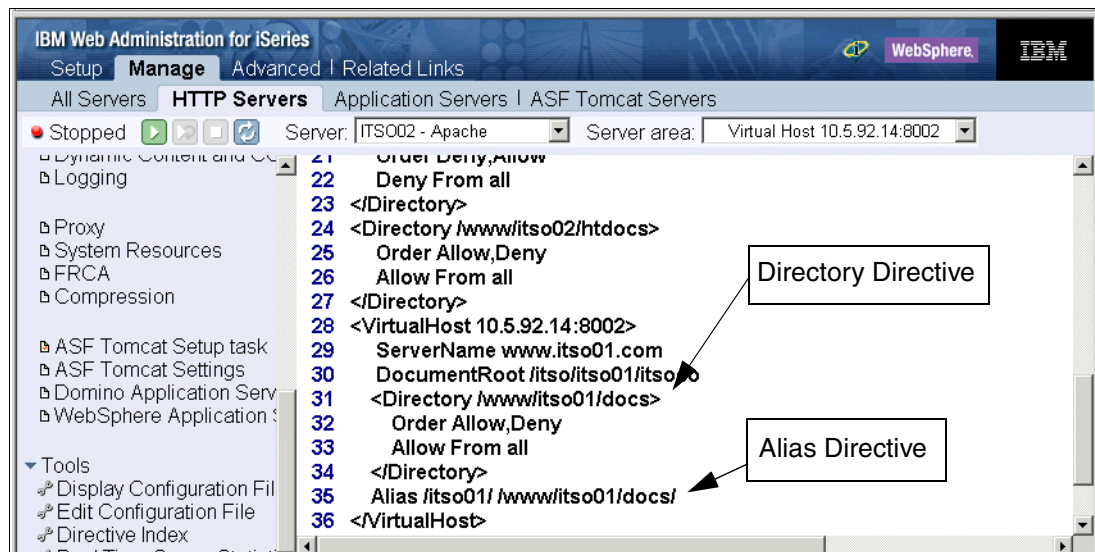


Figure 5-14 IP-based virtual hosting: Directory and alias directives

9. Create an error log file for the first virtual host as described in the following steps. For administration and problem determination, it is useful to have a different error log file for each virtual host. This approach may impact the server performance. See 10.2.3, “Logging” on page 230, for more information.
 - a. From the Server area list, select the server you want to work with. For this example, we select **Virtual Host 10.5.92.14:8002**.
 - b. From the left pane, under Server Properties, select **Logging**.
 - c. In the Logging panel (Figure 5-15), click the **Error Logs** tab.
 - d. From the Enable error logging list, select **Enabled**.
 - e. In the Error log field, type the error log file you want to use. For this example, we entered `/itso/itso01/logs/error_log`. If you want to change the error logging level, scroll down. From the Logging level list, select the appropriate logging level for your environment (not shown).

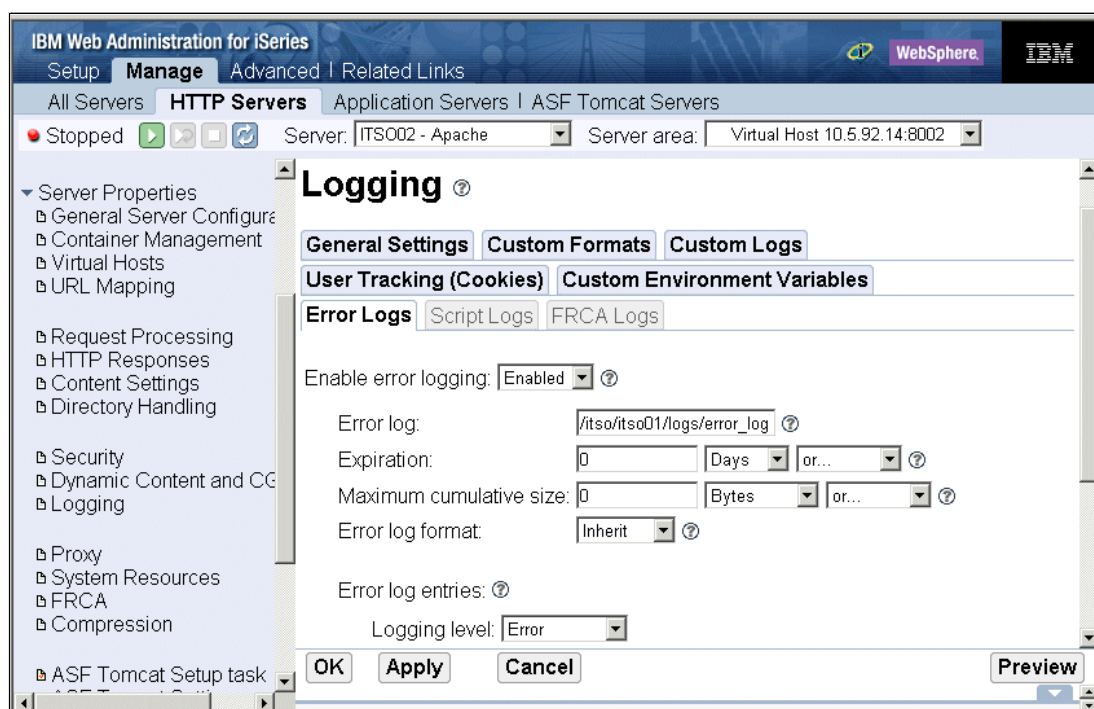


Figure 5-15 IP-based virtual hosting: Adding error logs

- f. Click **OK**.

Tip: For more information about the error logging level, see the HTTP online help.

Now, the HTTP configuration file has new entries named `ErrorLog` and `LogLevel` for each `<VirtualHost>` context. Figure 5-16 shows these new server directives. If you want to create a different error log file for each virtual host, repeat step 9.

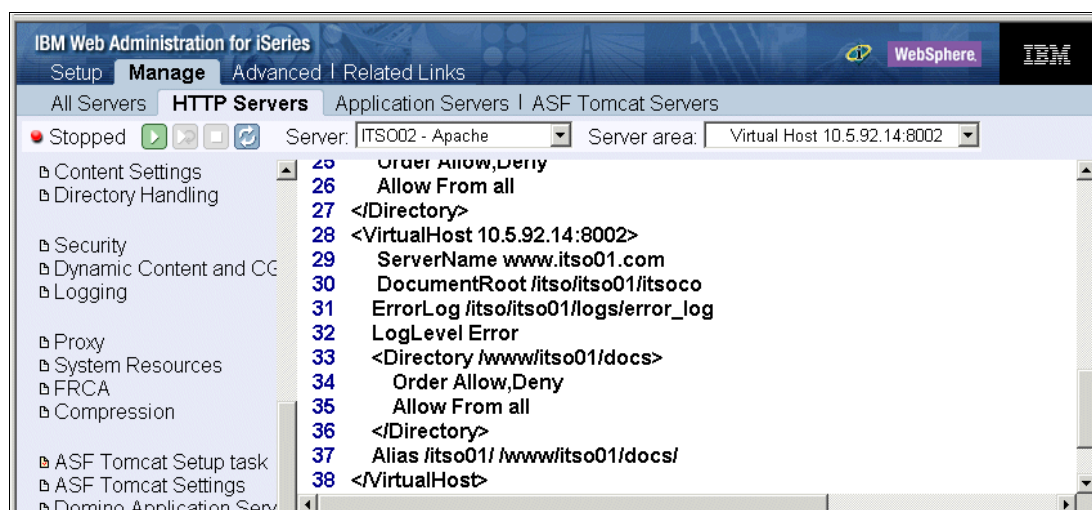


Figure 5-16 IP-based virtual hosting: Error logging directives

You created two `<VirtualHost>` contexts that will handle the incoming request to two different domains. Now, it is time to see how they work. Start your server instance and then test it as explained in the following steps.

1. Select the **Manage** tab.
2. From the Server list, select **ITS002**.
3. Click the **Start** icon, which is circled in Figure 5-17.

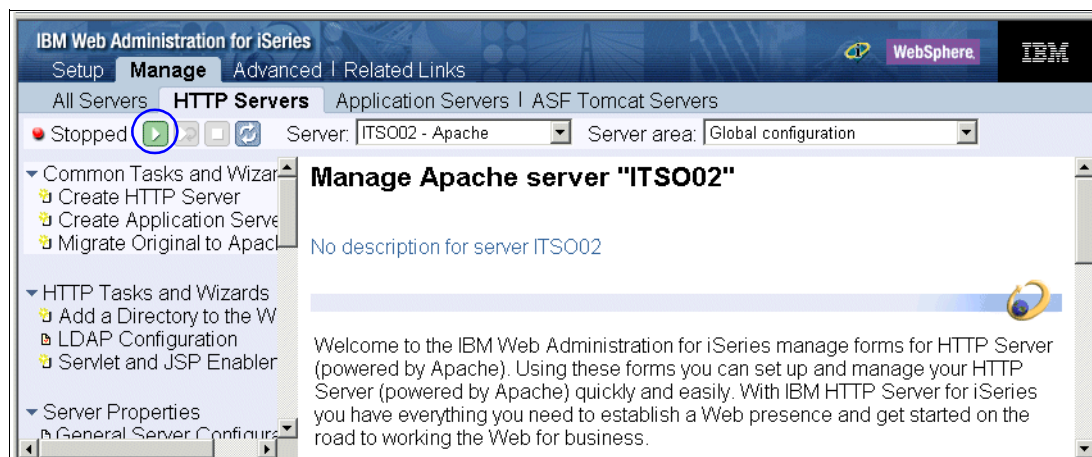


Figure 5-17 Starting the server instance

Tip: If you want to start the server instance with any specific startup parameter, you can enter the value in the *Server startup parameters* field. For example, you can use the `-t` option to verify the configuration file syntax, such as to find misspelled directory names. If there are errors, the command will notify you immediately.

4. To test the server instance, start a Web browser.
5. Type the URL for your domain. For this example, we type `http://10.5.92.14:8002` for the first virtual host.
6. Type the URL of another domain. For this example, we type `http://10.5.92.28:8002` for the second virtual host.

For both URLs, the `index.html` page should be displayed.

The IP-based implementation is simple, useful, and easy to configure, but it requires a dedicated IP address for each virtual host. Since IP addresses can be an expensive Internet resource, named-based virtual hosting was introduced.

5.4 Virtual hosts: Name-based implementation

The named-based virtual host implementation allows one IP address and TCP/IP port to host more than one domain. The benefits of using the name-based virtual hosts implementation is practically unlimited domains, ease of configuration and use, and no additional hardware or software resources required. Figure 5-18 shows a graphical representation for name-based virtual hosts.

Contrary to IP-based virtual host implementation, name-based virtual hosts rely on client Web browsers supported by the HTTP Version 1.1 protocol, specifically, the `hostname` information header. For name-based virtual hosting, all the Web clients must support HTTP 1.1 (or HTTP 1.0 with 1.1 extensions). The latest versions of most browsers support HTTP 1.1.

Simply stated, name-based virtual hosting requires that the client requests, which are being routed to the same physical interface with the same IP address, carry the host name in the HTTP headers so the HTTP server can distinguish between virtual hosts.

In addition to the `<VirtualHost>` directive used by IP-based implementation, the name-based virtual hosting uses the *NameVirtualHost* directive. This directive specifies an IP address (or host name that is mapped to an IP address) that should be used as a target for name-based virtual hosts as shown on Figure 5-19. Although `www.itso.com` can be the host name, we recommend that you always use an IP address for performance reasons. Any additional directive can (and should) be placed into the `<VirtualHost>` context.

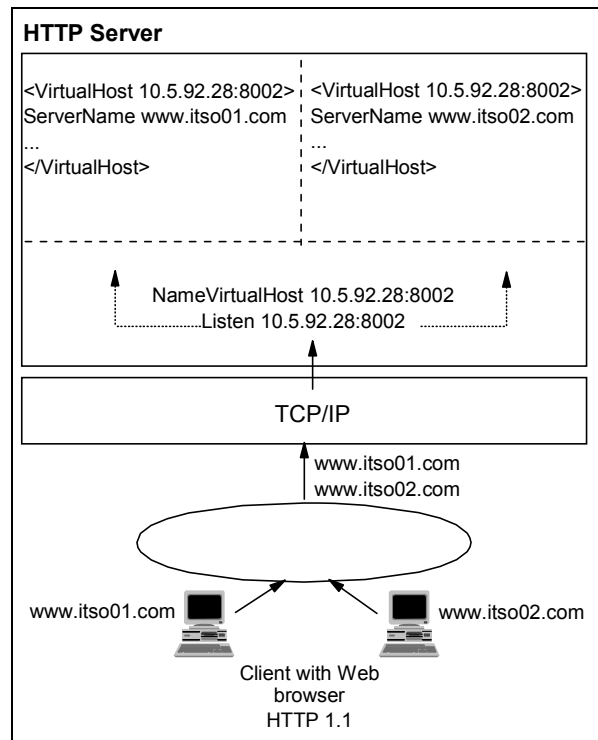


Figure 5-18 Named-based virtual host implementation example

The HTTP Server configuration name base looks like the example in Figure 5-19.

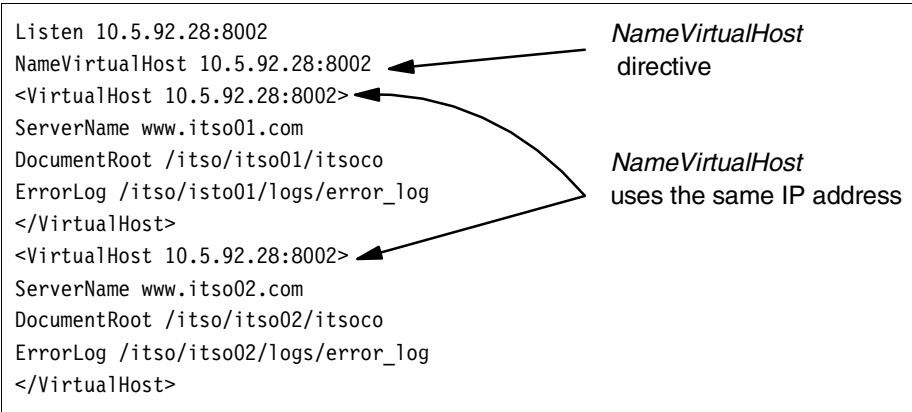


Figure 5-19 NameVirtualHost directive

With the name-based virtual host configuration, make sure the DNS or static host tables are configured so that one or more domains point to the same IP address. Otherwise, the requests are rejected.

5.4.1 Name-based virtual hosts: Problem overview

This time, your company needs to host two different domains, `www.itso01.com` and `www.itso02.com`, but your iSeries server only has one IP address to serve the incoming requests. Since the system only has one IP address, we decided to serve both domains using name-based virtual host implementation. This allows the HTTP server to handle the client request based on the domain name as shown in Figure 5-20. All the Web browser clients must support the HTTP 1.1 protocol.

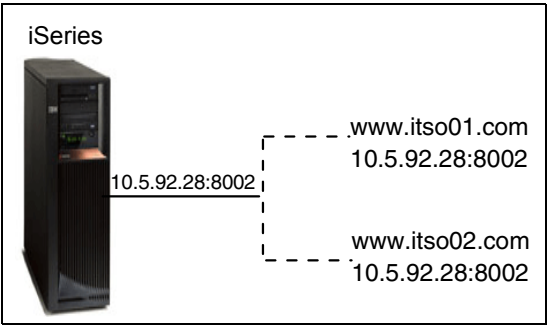


Figure 5-20 Name-based problem overview

To configure the HTTP Server (powered by Apache) to handle visitor requests, you must identify information related to the domain configuration as shown in Table 5-8. With that information, you can create the name virtual host configuration for the site. The Web server resources are the same as those used for the IP-based implementation (see Table 5-5 on page 79). But now, your iSeries server only has one IP address for all incoming client requests.

Table 5-8 Basic Web server resources

Resource	ITSO01	ITSO02
ServerName	www.itso01.com	www.itso02.com
Welcome page	index.html	index.html
IP address	10.5.92.28:8002	10.5.92.28:8002
DocumentRoot	/itso/itso01/itsoco	/itso/itso02/itsoco
ErrorLog	/itso/itso01/logs/error_log	/itso/itso02/logs/error_log

5.4.2 Name-based virtual host: Solution overview

The name-based virtual host configuration allows you to serve both domains using only one IP address. To accomplish this, you must use the HTTP server directive `NameVirtualHost`. This directive tells the HTTP server that you are going to share the IP address, but each request, depending on the domain name, must be handled independently. The configuration requires the following steps:

1. Add a name-based virtual host.
2. Include the IP address and port on which the server will listen.
3. Create the name-based virtual host context.
4. Populate the name-based virtual host context.

When the configuration is done, the HTTP configuration file includes the server directives shown in Table 5-9. We include the original server directives as a reference in case you have worked with the HTTP Server (original) before.

Table 5-9 Name-based overview

Original	Admin GUI steps	Apache final configuration file
<ul style="list-style-type: none"> ► DNS with entries that resolve the IP address ► Pass directives: Pass /itso01 /itso01/itsoco www.itso01.com Pass /itso02 /itso02/itsoco 	Add Listen directives for the IP address and port 2 .	1 # Configuration originally created by 2 Listen 10.5.92.28:8002 ...
	Add a name virtual host 18 .	18 NameVirtualHost 10.5.92.28:8002
	Create the virtual host context for each domain: ► One for www.itso01.com 19 . ► The other for www.itso02.com 33 .	19 <VirtualHost 10.5.92.28:8002> 20 DocumentRoot /itso/itso01/itsoco 21 ServerName www.itso01.com 22 UseCanonicalName Off 23 HostNameLookups off 24 ErrorLog /itso/itso01/logs/error_log 25 LogLevel error 26 <Directory /itso/itso01/itsoco> 27 AllowOverride None 28 order allow,deny 29 allow from all 30 </Directory> 31 Alias /itso01/ /itso/itso01/itsoco/ 32 </VirtualHost>
	Populate the virtual host context: ► Add server directives for the first domain 20, 21, 24, 25, 26, 31 . ► Add server directives for the second domain 34, 35, 35, 38, 39, 40, 45 .	33 <VirtualHost 10.5.92.28:8002> 34 DocumentRoot /itso/itso02/htdocs 35 ServerName www.itso02.com 36 UseCanonicalName Off 37 HostNameLookups off 38 ErrorLog /itso/itso02/logs/error_log 39 LogLevel error 40 <Directory /itso/itso02/itsoco> 41 AllowOverride None 42 order allow,deny 43 allow from all 44 </Directory> 45 Alias /itso02/ /itso/itso02/itsoco/ 46 </VirtualHost> ...

5.4.3 Name virtual host: Step-by-step implementation

The step-by-step configuration process is similar to the process we followed for the IP-based configuration. Here, there is an additional step to include for the new server directive `NameVirtualHost`.

Here are the global steps used to create the new name-based virtual host context. Follow the same configuration steps as demonstrated in 5.3.3, "IP-based virtual host: Step-by-step implementation" on page 80:

1. Start the Administrative GUI.
2. Select the HTTP server you want to work with. From the Server area list, select **Global Configuration**.
3. Include the IP address and port on which the server will listen. Under General Server Configuration, select the **General Settings** tab. Include the IP address and port for the virtual host.
4. Add a named virtual host. Under Virtual Hosts, select **Name-based**. Add the server name.
5. Create the name-based virtual host. Under Virtual Hosts select **Name-based**. Here you create the virtual host container for each domain. You also enter the server name and the document root.

For more server directives, see HTTP Server Documentation Center on the HTTP Server documentation Web site at:

<http://www-1.ibm.com/servers/eserver/series/software/http/docs/doc.htm>

Sometimes it is necessary to access the same virtual host by more than one server name. Additional names can be listed with the *ServerAlias* directive. For example, if you want to access `www.itso01.com` also with the name `itso01`, you can use the *ServerAlias* directive to accomplish this.

Here, we create an additional name for the name virtual host configuration we just created (the *ServerAlias* directive also can be configured using IP-based implementation). This time the HTTP configuration file has an additional entry as shown in Figure 5-21.

```
Listen 10.5.92.28:8002
NameVirtualHost 10.5.92.28:8002
<VirtualHost 10.5.92.28:8002>
  ServerName www.itso01.com
  ServerAlias itso01
  DocumentRoot /itso/itso01/itsoco
  ErrorLog /itso/itso01/logs/error_log
</VirtualHost>
<VirtualHost 10.5.92.28:8002>
  ServerName www.itso02.com
  DocumentRoot /itso/itso02/itsoco
  ErrorLog /itso/itso02/logs/error_log
</VirtualHost>
```

Figure 5-21 *ServerAlias* entry

To add the *ServerAlias* directive, complete the following steps.

1. From the Server area list, select the virtual host you want to work with. For this example, we select **Virtual Host 10.5.92.28:8002**.
2. From the left pane, under Server Properties, select **Virtual Hosts**.
3. In the Virtual Hosts panel (Figure 5-22), select the **Name-based** tab.
4. Under Additional names, click **Add** and type the new alias name. For this example, we enter `itso01`.
5. Click **Continue**.
6. Click **OK**.

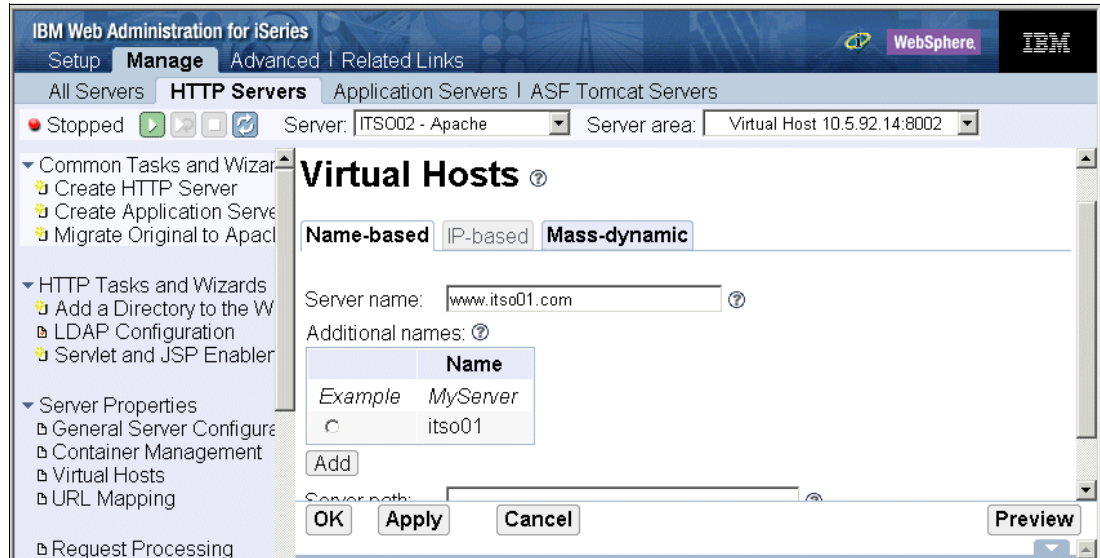


Figure 5-22 ServerAlias directive

7. Start the HTTP server instance and test the configuration. You can test the new ServerAlias directive by making a request to the Web site. This time use the server alias name, which is `http://itso01` for this example.

Tip: The IBM HTTP Server for iSeries Documentation Center offers a good explanation of the name-based virtual hosting including a sample configuration. Refer to the following Web site and select **e-business and Web serving** → **HTTP Server** → **Scenarios** → **Add virtual hosts**:

<http://publib.boulder.ibm.com/infocenter/iseries/v5r3/ic2924/index.htm?info/rzaie/rzaiemain.htm>

There are many configuration possibilities for the IP addresses and domain names your system is going to serve. For example, you may have one IP address with multiple domain names using a default port or specific port with the Listen directive, or multiple IP addresses with multiple domain names using default port or specific port, and so on. Although the configuration process is basically the same, there are some server directives you should include to resolve visitor requests aptly.

For more information about virtual host configurations, see the virtual host examples for common setups using HTTP Server (powered by Apache) Version 1.3. You can find them on the Web at:

<http://httpd.apache.org/docs/vhosts/examples.html>

5.5 Virtual hosts: Mass dynamic implementation

Mass dynamic virtual host implementation allows you to add dynamically domains (host names) by adding directories of content. This approach is based on automatically inserting the IP address (or host name) and the content of the Host:

header into the path name of the file that is used to satisfy the request. This

means that using the host name provided in the URL requested by the client, the HTTP server processes the request as shown in Figure 5-23.

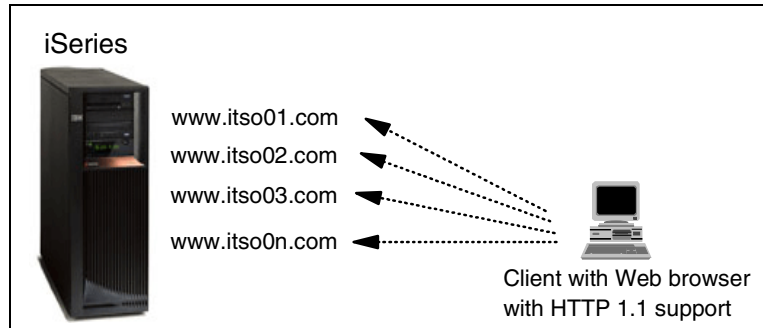


Figure 5-23 Mass dynamic implementation

The mass dynamic virtual host implementation differs from IP-based or name-based in the mechanism used to determine the location of the files you want to serve. Here, the HTTP server uses the content that the host provides in the URL to serve visitor requests. Basically, mass dynamic virtual host uses a variable path name (based on the header) to find the file system structure of the static data that the site is going to serve. Using a mapping mechanism and the mass dynamic virtual host, the HTTP server converts:

- ▶ `http://www.itso01.com` into `/itso/itso01/itsoco`
- ▶ `http://www.itso02.com` into `/itso/itso02/itsoco`

The conversion process is supported by specifiers inspired by the UNIX command **printf**, which has a number of formats as shown in Table 5-10.

Table 5-10 UNIX printf specifiers

Variable	Value
%%	Insert a %
%p	Insert the port number of the virtual host
%N.M*	Insert (part of) the name
* N and M are used to specify substrings of the name. N selects from the dot-separate component of the name, and M selects characters within whatever N has selected. M is optional and defaults to zero if it is not present. The dot must be present if and only if M is present.	

The client request is processed based on the URL. Which part retrieves the HTTP server depends on the value you write in the mass dynamic virtual host directives using the information in Table 5-11.

Table 5-11 Mass dynamic value interpretation

Value	Description
0	The whole name
1	The first part
2	The second part
-1	The last part
-2	The next to the last part

Value	Description
2+	The second and all subsequent parts
-2+	The next to last part and all preceding parts
1+ and -1+	The same as 0

Using the specifiers in the Table 5-10 and the values in the Table 5-11, mass dynamic performs the interpretation process, called *directory name interpolation*. The interpolation process requires that the interpolated directory exists into the file system since the Web server name is translated into physical path names in the iSeries integrated file system (IFS). For example, if the domain name `www.itso01.com` is interpolated into `/itso/itso01/itsoco`, the directory `/itso/itso01/itsoco` *must* exist in the IFS. Otherwise, the request fails.

Mass dynamic implementation is supported by the `mod_vhost_alias` module. This module supports the server directives associated with the mass dynamic host implementation. The directives are:

- ▶ **VirtualDocumentRoot:** Allows you to determine where the server looks for the document root based on the value of the server name
- ▶ **VirtualDocumentRootIP:** Allows you to determine where the server looks for the document root based on the IP address
- ▶ **VirtualScriptAlias:** Allows you to specify the directory path where the server looks for CGI scripts based on the value of the server name
- ▶ **VirtualScripAliasIP:** Allows you to specify the directory path where the server looks for CGI scripts based on the IP address

5.5.1 Mass dynamic virtual host: Problem scenario

Using one HTTP server to host multiple domains becomes inefficient if the HTTP configuration file contains many `<VirtualHost>` contexts that are substantially the same. The following example illustrates this situation:

```
Listen 10.5.92.28:8002
NameVirtualHost 10.5.92.28:8002

<VirtualHost 10.5.92.28:8002>
    ServerName      www.itso-01.com
    DocumentRoot    /itso/www.itso-01.com/itsoco
    ScriptAlias      /cgi-bin/ /itso/www.itso-01.com/itsoco/cgi-bin
</VirtualHost>

<VirtualHost 10.5.92.28:8002>
    ServerName      www.itso-02.com
    DocumentRoot    /itso/www.itso-02.com/itsoco
    ScriptAlias      /cgi-bin/ /itso/www.itso-02.com/itsoco/cgi-bin
</VirtualHost>
# and so on...
<VirtualHost 10.5.92.28:8002>
    ServerName      www.itso-0n.com
    DocumentRoot    /itso/www.itso-0n.com/itsoco
    ScriptAlias      /cgi-bin/ /itso/www.itso-0n.com/cgi-bin
</VirtualHost>
```

This HTTP server is hosting multiple domains using the name-based implementation. Here, every <VirtualHost> context has a DocumentRoot and ScriptAlias related to the value in the ServerName directive. With the advantages of the mass dynamic virtual host, we are going to interpret the domain name. Based on the interpretation, the HTTP server processes the request. Using this new implementation, the HTTP configuration file looks like the following example:

```
Listen 10.5.92.28:8002
NameVirtualHost 10.5.92.28:8002
UseCanonicalName Off
...
VirtualDocumentRoot /www/%2/itsoco
VirtualScriptAlias /itso/%0/itsoco/cgi-bin
```

In the new configuration file, there is no ServerName directive, because this ServerName is provided by the URL received in the client request. The way the HTTP server identifies the ServerName provided in the header is based on the value configured to the UseCanonicalName directive as shown in Table 5-12.

Table 5-12 UseCanonicalName directive

UseCanonicalName value	Use
Off	The HTTP server forms a self-referential URL using the host name and port supplied by the client.
DNS	The HTTP server does a reverse DNS lookup on the server IP address that the client connected to in order to work out a self-referential URL.
On	The HTTP server uses the ServerName and Port directives to construct a canonical name for the server.
Not include	The HTTP server uses the TCP/IP Domain of the server.

The advantages of the mass dynamic implementation are:

- ▶ It adds domains dynamically.
- ▶ You do not need to restart the HTTP server to serve a new domain.

The disadvantages of this implementation are:

- ▶ There are no individual logs when used with IP or named virtual host implementations.
- ▶ There is no tailoring of individuals domains with use of other directives in a virtual host context.

5.5.2 Mass dynamic virtual host: Solution overview

To understand the advantages of the mass dynamic virtual host, we are going to act as an ISP. Using the iSeries server and the HTTP Server (powered by Apache), we are going to create an HTTP server required to host domains dynamically. We need to include the mass dynamic directives that allow us to process the request for the following domain names:

- ▶ www.itso01.com
- ▶ www.itso02.com
- ▶ www.itso0n.com

We need to find the appropriate interpolation value that allows us to use the header provided in the URL, retrieve the host name, and process the request. We perform these steps:

1. Retrieve the second part of the host name provided in the URL. In our case, this is the host name.
2. Interpolate the host name into some directory that exists in the iSeries IFS.
3. Serve the documents from that IFS directory.

In our example, the second part `itsoXX` is part of the document root directive as shown in Figure 5-24.

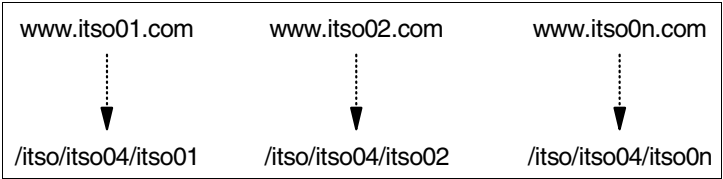


Figure 5-24 Mass dynamic problem overview

Using the interpolation values and the mass dynamic directives, we must include the following directive in the HTTP configuration:

```
VirtualDocumenRoot /itso/itso04/%2
```

Here `/itso/itso04` is the document root of the HTTP server. Also, `/%2` retrieves the second part of the URL request. It is the directory used to process the requests and the place where the HTML code, images, and so on are located.

The mass dynamic implementation requires some configuration. Some configuration steps and their results are shown in Table 5-13. Note that the HTTP Server (original) really has no equivalent function or feature.

Table 5-13 Mass dynamic overview

Apache GUI steps	Apache final configuration file
Add Listen for the IP address 3.	2 LoadModule vhost_alias_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
Add the mass dynamic directives 2 and 22.	3 Listen 10.5.92.14:8004
Include the UseCanonicalName directive 9	... 9 UseCanonicalName Off ... 22 VirtualDocumentRoot /itso/itso04/%2 23 <Directory /> 24 AllowOverride None 25 order deny,allow 26 deny from all 27 </Directory> 28 <Directory /itso/itso04> 29 AllowOverride None 30 order allow,deny 31 allow from all 32 </Directory>

The following section includes the step-by-step configuration options. Before you create the mass dynamic configuration, you must identify the specifiers that will interpolate your site `ServerName` into the directory structure.

5.5.3 Mass dynamic virtual host: Step-by-step implementation

To create the HTTP Server (powered by Apache) mass dynamic configuration, follow these steps:

1. Create the HTTP server.
2. To create the mass dynamic entries, select your server from the Server list. We selected **ITSO04**. From the Server area list, select **Global Configuration**.
3. In the left pane, under Server Properties, select **Virtual Hosts**.
4. In the Virtual Hosts panel (Figure 5-25), click the **Mass-dynamic** tab.
5. For the How to build a self-referencing URL option, select **Do not build self-referencing URLs - use hostname and port supplied by client**.

Remember that this option determines how a URL is constructed. Using the selected option, the server constructs self-referencing URLs by using the host name and port that was provided by the user in the browser.

6. In the Root directory for serving files field, type the virtual document root. For this example, we enter `/itso/itso04/%2`.
7. Click **OK**.

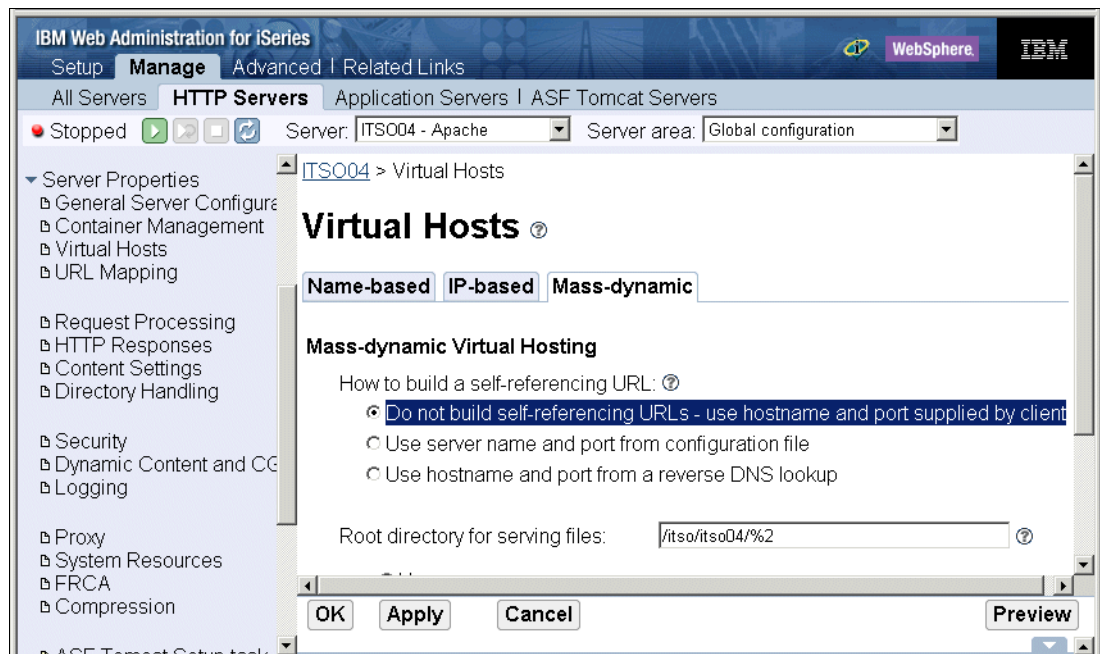


Figure 5-25 Mass dynamic configuration

Now, the HTTP configuration file has new entries whose numbers are circled in Figure 5-26. These entries load the module required to handle the mass dynamic request and the HTTP server directive that is required to interpolate the request.

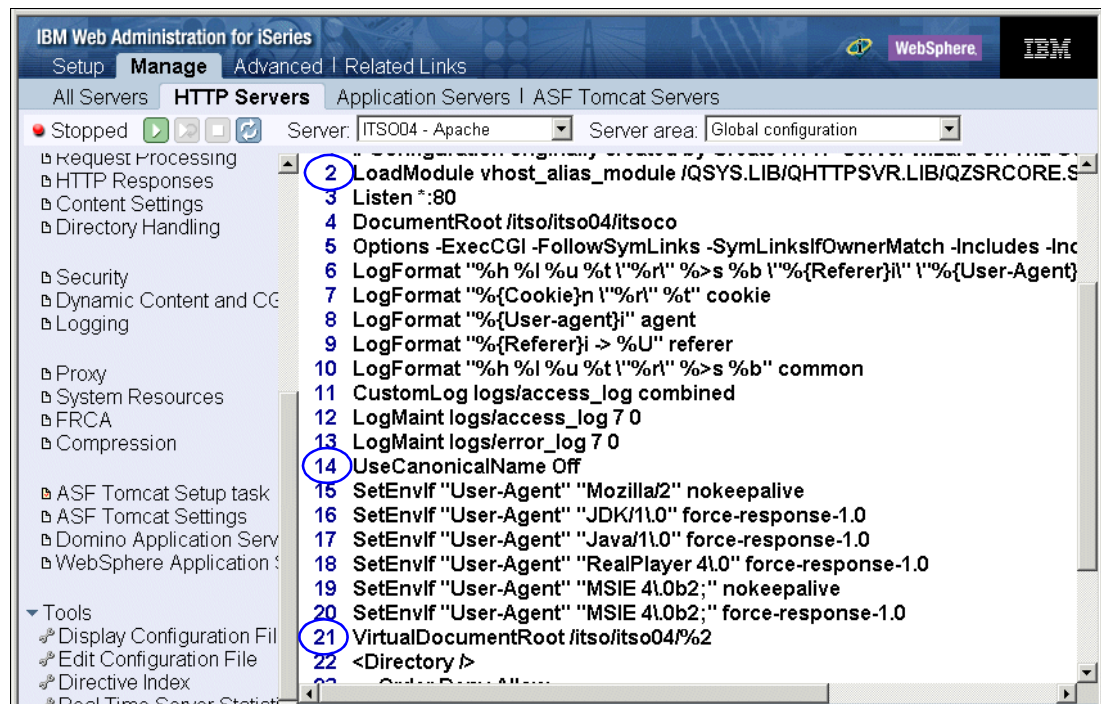


Figure 5-26 Mass dynamic module and virtual document root directive

The mass dynamic configuration is ready. To test the configuration, start the HTTP server and open a client Web browser. In our scenario, a URL of `http://www.itso03.com:8004/` results in the display shown in Figure 5-27.

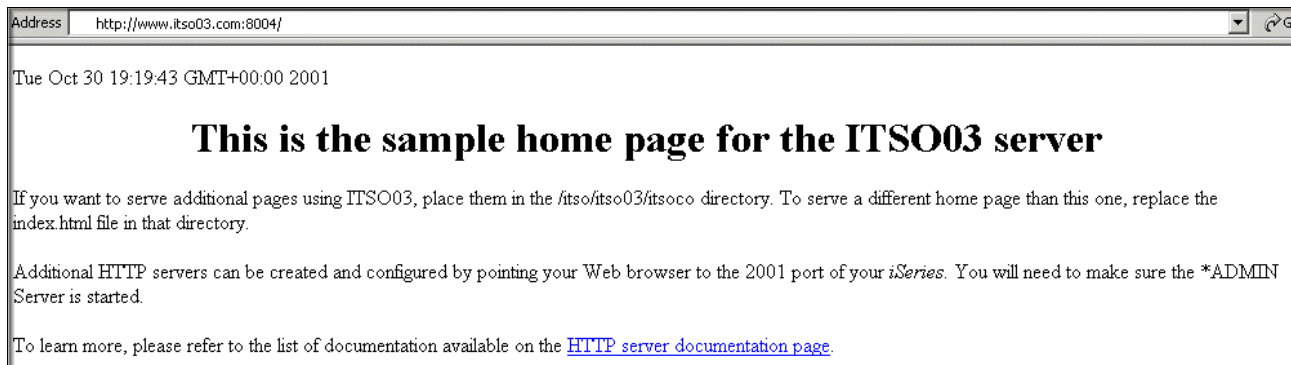


Figure 5-27 Mass dynamic example: URL `http://www.itso03.com:8004/`

A URL of `http://www.itso02.com:8004/` results in the display shown in Figure 5-28.

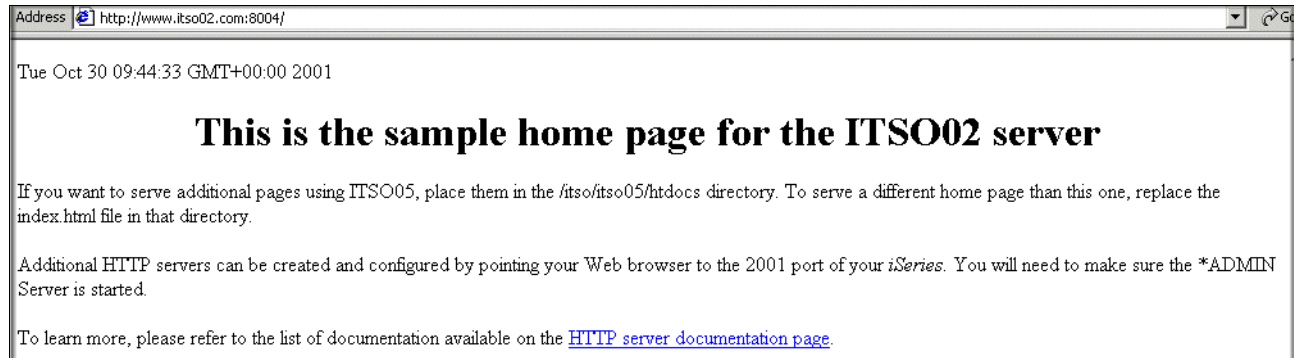


Figure 5-28 Mass dynamic example: URL `http://www.itso02.com:8004/`

The advantages of a dynamic virtual host are:

- ▶ A smaller HTTP configuration file so the server instance starts faster and uses less memory
- ▶ Easy administration since adding virtual host does not require configuration changes or server restarts

The mass dynamic example that we showed was based on a URL. However, you can create a mass dynamic configuration based on the IP address. To do this, you simply turn the `UseCanonicalName` from *off* to *DNS* and use the mass dynamic virtual host directives related to the IP, such as `VirtualDocumentRootIP`.

Defending the IFS

Security is always a main concern on the mind of a Web server administrator. Even though your server only runs on a private intranet, you should not underestimate the importance of security planning. Private networks are not exempt from security exposures, as recent waves of Internet worms that have made their way into intranets have repeatedly proven.

Security comes from a set of constantly updated rules and practices, specifically designed to protect the availability of your server and the integrity of your data. Figure 6-1 presents a high-level overview of iSeries server security in a network environment. A network security layer, encompassing both physical devices and software filters, is the outer protector of your iSeries fortress.

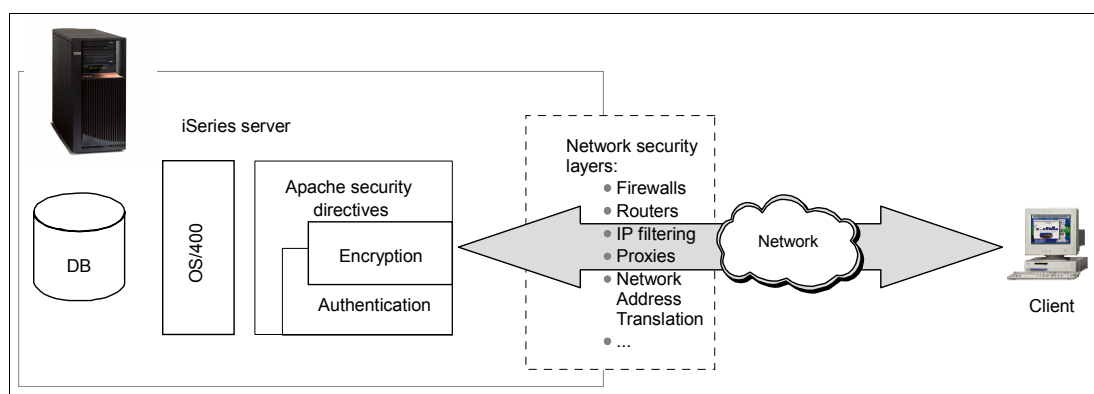


Figure 6-1 iSeries security in the network environment

Once inside, all requests are filtered by Apache server security. Data is protected through:

- ▶ **User authentication:** The process of verifying a user's identity through some sort of credentials. This can either be done through user ID and password combinations or through an exchange of digital keys (or certificates).
- ▶ **Access control:** Specifically, at this point, we discuss access control from the HTTP Server (powered by Apache)'s point of view, above the access control that OS/400 also enforces. This is enforced through a set of policies that define who can access your data, what kind of authority they are granted, and what actions they are allowed to perform on

your data. A server-wide access control policy is enforced on the document root and propagated upon lower-level contexts unless overridden by local directives or local configuration files. In addition to that, the server never tries to access system resources for which explicit access has not been configured.

- ▶ **Encryption:** A mathematical process is used to disguise data to keep unauthorized parties from gaining access to sensitive information. Data is encrypted into ciphertext using a unique key and a set of operations that define an algorithm. The strength and effectiveness of encryption techniques depend on three factors: complexity of the algorithm, the length of the encryption key, and the overall strength of the key itself. A compromised key can easily render the strongest encryption techniques completely useless.

At the core of your system, the renowned strength of OS/400 security is the ultimate defender of your database and all objects on your server. The main reason is that the HTTP Server (powered by Apache) is just another job running under the operating system. If the user profile associated with the instance of the HTTP Server (powered by Apache) or the user profile associated with an authenticated user does not have access to the object, then *nobody* can access that object.

6.1 Access control

Apache enforces access control through configuration directives at server or virtual host level. The control policies are inherited by lower-level contexts and can only be overridden by either local configuration directives or access control files (see the following paragraph). The following directives define access control policies:

- ▶ **Allow:** Specifies which client hosts are allowed access to server resources
- ▶ **Deny:** Specifies which client hosts are not allowed access to server resources
- ▶ **Order:** Controls the order in which deny and allow directives are evaluated
- ▶ **Require:** Indicates which users and groups are allowed access to server resources

Access control files are usually named `.htaccess`, but you can change the name using the `AccessFileName` directive. If used together with `AllowOverride` directive, `.htaccess` files can define context-based configurations. These files are not parsed at startup, but every time a request is processed. This means that changes to those files do not require a server restart. You should also be aware that the server looks for `.htaccess` files in every accessible directory and subdirectory before it serves a request.

Tip: In general, you should configure `AllowOverride None` as your root or default behavior. This causes your HTTP Server (powered by Apache) to not even look for the `.htaccess` file.

Why? One reason is performance. For every access to your system, OS/400 must perform extra input/output (I/O) to look for the `.htaccess` file (even if it is not there), which decreases performance. Another reason is security. Setting `AllowOverride None` gives you an extra padding of security. For example, if a hacker can place their own `.htaccess` file (with security-related directives within), they can immediately open your server for attack. This also applies to local OS/400 user profiles. In many installations, OS/400 user authority to directories, such as your directories you serve content from, is too high.

This also allows a signed on user to place or modify `.htaccess` files in a directory. One place to use the `.htaccess` file is when you need distributed administration and configuration. See 4.1, “In-context configuration” on page 60, for a wider discussion of `.htaccess`.

The Order directive is sometimes confusing to new Apache administrators. It is worth spending your time reading Apache text books, Web sites, or help text via the HTTP Server (powered by Apache) to understand how it works. For example, you may have a configuration file such as:

Even the Order directive is implemented as a series of default behaviors that are later overridden.

```
Order deny,allow
    Deny from all
    Allow from somehost
```

This means (in order of precedence):

1. Allow is the *default* and rules if no Deny or Allow was specified.
2. To deny, you *must* explicitly specify the clients that you want to deny. In this example, we are saying that we want to *Deny* all clients (by Internet Protocol (IP) address).
3. Allow *overrides* the Denied clients. In this case, the client coming from the IP address behind somehost is the only client allowed to this directory.

Tip: You must enter the keyword *deny,allow* or *allow,deny* exactly as shown. No space is found between the two words since this is seen by the Apache server as a single keyword.

6.2 Basic authentication

Basic authentication is a popular means of verifying a user's identity before granting access to a protected resource or *realm*. Figure 6-2 illustrates the authentication process. The process flow is explained here (note that the step numbers correspond to those in Figure 6-2):

1. The client requests access to a protected resource.
2. The server replies with HTTP status code 401 (see 13.2.8, "HTTP status codes" on page 352) and a special header, WWW-Authenticate, that contains the name of the protection realm.
3. The client interprets the WWW-Authenticate and presents the user with a login prompt, requesting valid credentials for the realm.
4. The user's credentials are sent back to the server for validation.

Important: If the authentication request is sent via an HTTP request, and not an HTTPS request, the user credentials are only encoded, but not encrypted. The encoding is done via Base64 encoding and can be decoded easily through freely available decoding programs. We strongly recommend that you use a Secure Sockets Layer (SSL) protected (HTTPS protocol) session for authentication.

5. Depending on the method you choose, the credentials are checked against OS/400 user profiles, a validation list, or Lightweight Directory Access Protocol (LDAP) entries.
6. If the user's credentials can be verified, the client is granted access to the protected resource. Otherwise, an error message is returned in the browser window.

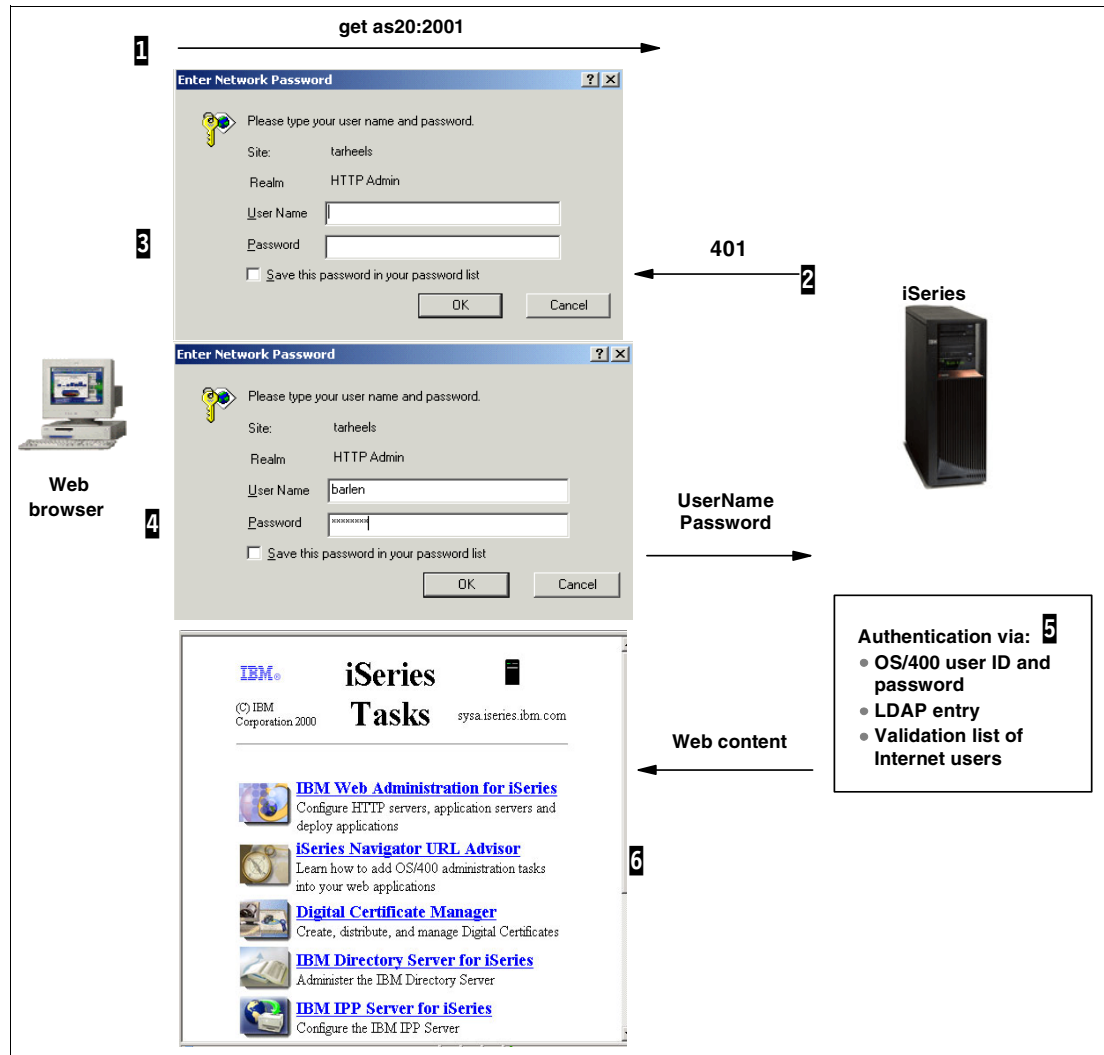


Figure 6-2 The 401 challenge process

Table 6-1 introduces the three different types of basic authentication that are available on the HTTP Server (powered by Apache):

- ▶ See 6.2.1, "Authentication by OS/400 user profiles" on page 105
- ▶ See 6.2.2, "Authentication by a validation list" on page 108
- ▶ See 6.2.3, "Authentication by LDAP entries" on page 113

We detail the steps to configure the three different authentication methods in the sections that follow. In the end, all three methods have more in common than they have differences. Table 6-1 demonstrates this. This table reads top down. By following your goals, you can use the graphical user interface (GUI) configuration steps to create the Apache final configuration file directives as listed.

You must have the directives on lines **34**, **36**, **38**, and **43**. Your choices within these configuration directive lines are common regardless of the basic authentication goal you choose. That is, you can change the name of the realm defined by `AuthName` to something other than `MyRealm`, but this does not affect your goal of basic authentication.

The directives for lines **41** and **42** in Table 6-1 allow you to make choices for either a user access policy and a user validation policy.

Table 6-1 Getting started with basic authentication

Goal	GUI configuration steps	Apache final configuration file directives
Locate and select the context to protect 34	Select a context from the menu on the left.	34 <Directory /ITSO/itso06/itsoco/Projects/Archives>
Create a protection setup 36	Choose Authentication name or realm	35 AllowOverride None 36 AuthName MyRealm 37 ProfileToken off 38 AuthType Basic 39 order allow,deny 40 allow from all
Choose a user access policy. The server will access this resource as: <ul style="list-style-type: none"> ▶ The webmaster (QTMHHTTP) 41a ▶ The validated user profile 41b ▶ A specific user profile 41c 	Select a User name to process requests from the list, or enter one of your choice. The server always swaps to this profile when serving content from this resource. Select Default server profile .	41a UserID %SERVER%
	Select User profile of the client .	41b UserID %CLIENT%
	Select - Other - and enter a valid OS/400 profile in the blank field	41c UserID gbanchelli
Choose a user validation policy. User credentials can be validated by: <ul style="list-style-type: none"> ▶ Validation list 42a ▶ OS/400 user profiles 42b ▶ LDAP entries 42c 	Select one of the options using the radio buttons. Go to 6.2.2, "Authentication by a validation list" on page 108, for details.	42a PasswdFile qgpl/itso06
	Go to 6.2.1, "Authentication by OS/400 user profiles" on page 105, for details.	42b PasswdFile %SYSTEM%
	Go to 6.2.3, "Authentication by LDAP entries" on page 113, for details.	42c PasswdFile %LDAP%
Enforce security 43	Under Authentication and Security, select Control Access . Select All authenticated users and click Apply .	43 require valid-user 44 </Directory>

6.2.1 Authentication by OS/400 user profiles

OS/400 user profiles can be used for authentication. The advantage of this implementation is that it does not require you to perform additional configuration steps or to maintain a separate user database. User profiles with limited capabilities and no signon access, and *SECOFR class users (although this practice is highly discouraged), can be used for this purpose.

Tip: Access validation through OS/400 user profiles is the simplest and, under certain circumstances, least secure way to restrict access to your data. While acceptable in non-critical environments, we do not recommend this kind of authentication alone on public networks such as the Internet, where its simple Base64 encoding and the use of actual user profiles and passwords can compromise the security of your system. A good choice for protecting your data is to use SSL. See 6.4, "Encrypting your data with SSL and TLS" on page 127, for an example.

Implementation

Follow these steps to create a configuration that authenticates a remote user by using OS/400's user IDs and passwords as shown in Figure 6-3:

1. From the Server area list, select the context that you will protect.
2. In the left pane, under Server Properties, select **Security**.
3. In the right panel, select the **Authentication** tab.
4. On the Authentication page, complete these steps:
 - a. Select **OS/400 user profiles**.
 - b. Specify a significant name for this realm. It's not only a unique identifier for this protection setup, but also a hint for the user to identify the type of authentication enforced.
 - c. Select the client authority that will perform access to this resource:
 - **Enabled:** When accessing the resource, the server temporarily switches to the user profile of the authenticated user and performs access under this user profile. A special value `%%CLIENT%%` is used on the UserID directive. The UserID directive overrides the ServerUserID directive.
 - **Disabled:** Access to the protected resources is, by default, performed under the QTMHHTTP profile for static pages and QTMHHTTP1 profile for CGI programs unless the ServerUserID directive specifically names another user profile.
 - d. If Disabled is selected for the client authority, enter the profile name that the server will use to access it. You must enter a user profile name. No special values are allowed.
 - e. Click **Apply** to save your settings.

Tip: When dealing with OS/400 user profiles, remember that users with *ALLOBJ authority are not subject to any access restriction on the file system, not even an explicit *EXCLUDE.

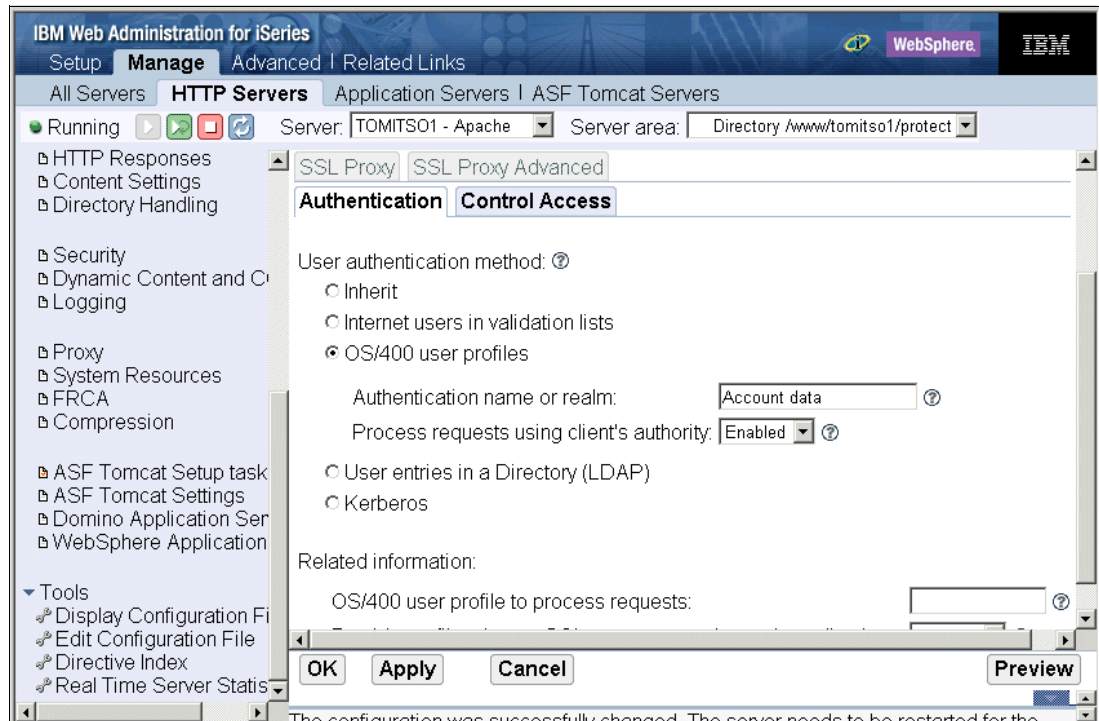


Figure 6-3 Basic Authentication

5. Select the **Control Access** tab (Figure 6-4).
6. On the Control Access page, complete these steps:
 - a. Select **All authenticated users (valid user name and password)**. This adds the require valid-user directive to your configuration and enforces the access restriction.
 - b. Click **Apply** to save your settings.

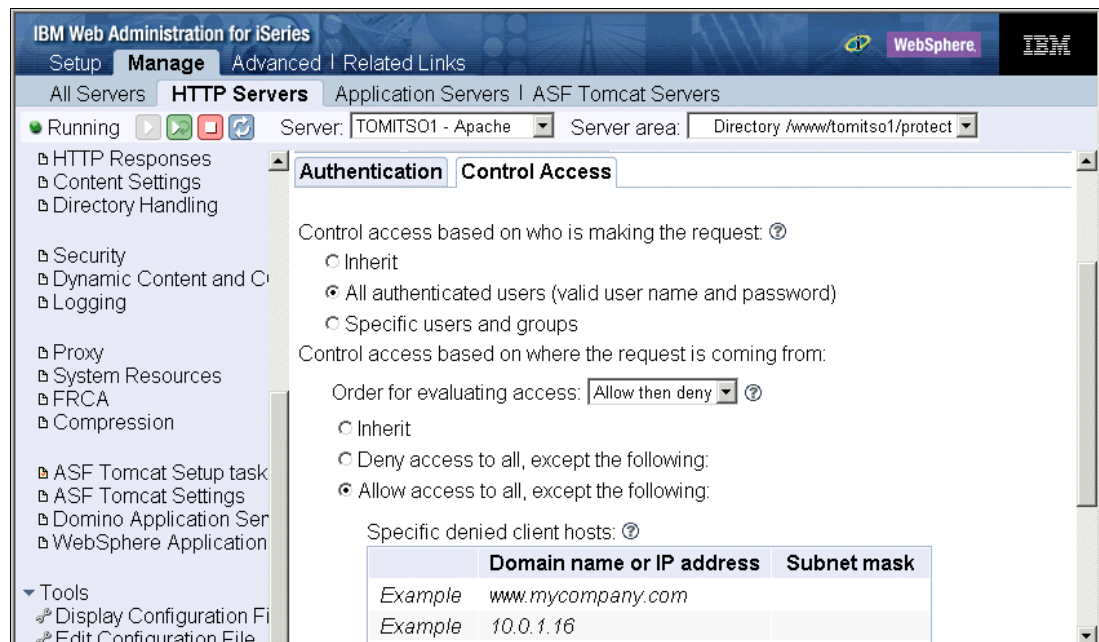


Figure 6-4 Control Access

Note: You can further grant access based on the IP address from which the request originates. In this case, you have to select the options in the *Control access based on where the request is coming from* and the *Control access policy* sections. Selections in these sections must also be made when you inherit settings from a parent context.

- c. Click **OK** to close the Security page.
7. Restart your server instance. Point your Web browser to the context that you just protected. You are then prompted for an OS/400 user name and password.

6.2.2 Authentication by a validation list

Protection by a validation list does not require the use of actual OS/400 profiles and passwords, reducing risk to your iSeries server in the event that a user ID is compromised. Like all other forms of basic authentication, passwords are sent Base64 encoded. That is, they are sent “in the clear”.

Implementation

First create a validation list for your Internet or intranet users. You use the Create Validation List (CRTVLDL) command as shown in Figure 6-5.

Create Validation List (CRTVLDL)

Type choices, press Enter.

Validation list	WEBUSERS	Name
Library	QGPL	Name, *CURLIB
Text 'description'	WebUsers protection realm	

Additional Parameters

Authority	*EXCLUDE	Name, *EXCLUDE, *USE...
---------------------	----------	-------------------------

Bottom

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

Figure 6-5 Create Validation List display

The default public authority on our newly created validation list is set to *EXCLUDE. While this is good for security, it also prevents the webmaster from accessing it for authentication purposes. Use the Grant Object Authority (GRTOBJAUT) or the Edit Object Authority (EDTOBJAUT) CL commands to grant the webmaster *CHANGE authority for the validation list as shown in Figure 6-6. This authority is needed to manage validation list users. The user profile under which the HTTP server runs needs *USE authority to the validation list to perform the user authentication.

```

                                Edit Object Authority (EDTOBJAUT)

Type choices, press Enter.

Object . . . . . > WEBUSERS      Name
Library . . . . . >  QGPL        Name, *LIBL, *CURLIB
Object type . . . . . > *VLDL    *ALRTBL, *AUTL, *BNDDIR...

                                                                    Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
Function key not allowed.

```

Figure 6-6 Edit Object Authority

Figure 6-7 illustrates the final EDTOBJAUT display after authority on the object is successfully changed. Notice that the *ALL authority given to the original owner is no longer needed once the list is set up.

```

                                Edit Object Authority

Object . . . . . : WEBUSERS      Owner . . . . . : BARLEN
Library . . . . . :  QGPL        Primary group . . . : *NONE
Object type . . . . : *VLDL

Type changes to current authorities, press Enter.

Object secured by authorization list . . . . . *NONE

User      Group      Object
BARLEN    Group      Authority
WEBMAST1  Group      *ALL
QTMHHTTP  Group      *CHANGE
*PUBLIC   Group      *USE
           Group      *EXCLUDE

                                                                    Bottom
F3=Exit  F5=Refresh  F6=Add new users  F10=Grant with reference object
F11=Display detail object authorities  F12=Cancel  F17=Top  F18=Bottom

```

Figure 6-7 Checking webmaster authority

Now add users to this validation list using the GUI as shown in Figure 6-8:

1. Click the **Advanced** tab and then the **Internet Users and Groups** subtab.
2. In the left pane, under Internet Users and Groups, select **Add Internet User**.
3. In the Add Internet User panel, complete the fields as shown in Figure 6-8 and click **Apply**.

The screenshot shows the 'IBM Web Administration for iSeries' interface. The top navigation bar includes 'Setup | Manage | **Advanced** | Related Links'. Below this, a subtab 'Internet Users and Groups' is selected. The left sidebar contains a tree view with 'Common Tasks and Wizards' and 'Internet Users and Groups'. Under 'Internet Users and Groups', 'Add Internet User' is selected. The main content area is titled 'Add Internet user' and contains the following fields:

- User name:**
- Password:**
- Confirm password:**
- Comments:**
- Validation list:**
- Group file:**
- Group:**

At the bottom of the form are 'Apply' and 'Reset' buttons.

Figure 6-8 Add Internet User

4. As shown in Figure 6-9, click the **Manage** tab and then the **HTTP Servers** tab.
5. From the Server area list, select the context you will protect.
6. In the left pane, select **Security**.
7. In the right panel, select the **Authentication** tab.
8. On the Authentication page, complete these tasks:
 - a. Under User authentication method, select **Internet users in validation lists**.
 - b. Enter a realm for the browser authentication prompt.
 - c. Under Validation lists, click **Add**.
 - d. In the Validation list table, enter the library and validation list name.
 - e. Click **Continue**.



Figure 6-9 Authentication by Validation List: Basic authentication

- f. Scroll down and specify a user profile to process requests as shown in Figure 6-10.

Specify a user name to process requests. In the case where you are using a validation list to verify the identity of the remote user, the authenticated user has no connection to an OS/400 user profile. Therefore, you cannot process the request under the client authority. However, you can enter a user profile name for OS/400 user profile to process requests. This user must have proper authority to the protected resources. If this parameter is left blank, the default server profile QTMHHTTP is used to access static content and QTMHHTTP1 for CGI programs.

Click **Apply**.

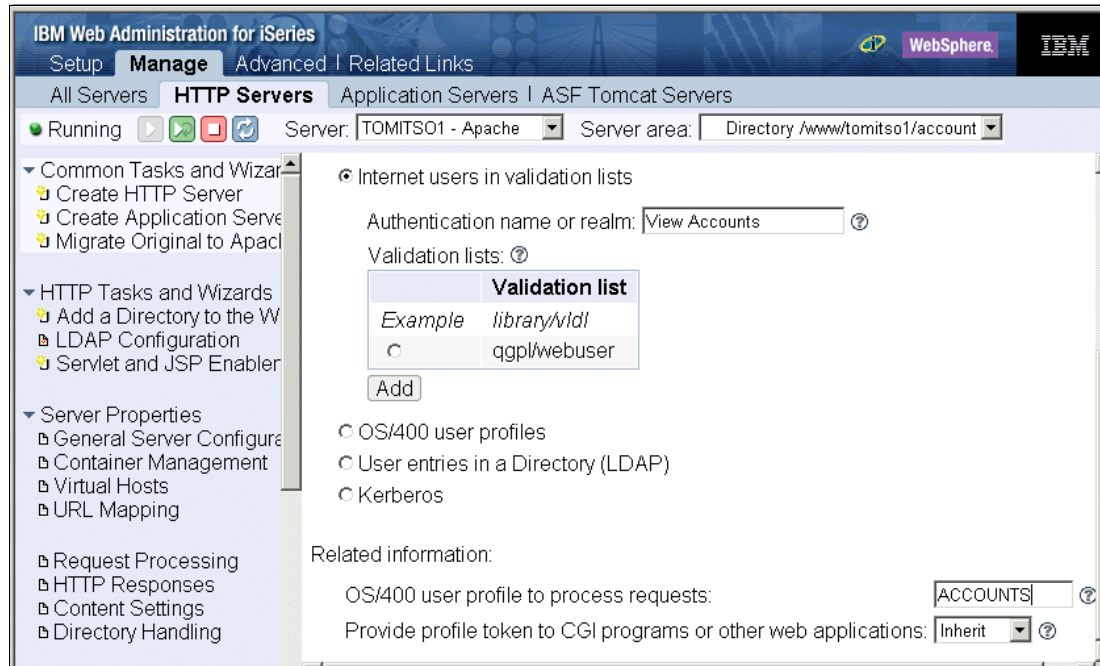


Figure 6-10 Authentication by Validation List: Specifying a user name

9. In the Security panel (Figure 6-11), select the **Control Access** tab.
10. On the Control Access page, complete these tasks:
 - a. Select **All authenticated users (valid user name and password)**.
 - b. Click **OK** to save the configuration. This activates the protection setup.
11. Restart your server instance. Point your Web browser to the context you just protected. You are then prompted for a user name and password. Use the one that you can entered into the validation list.



Figure 6-11 Control Access

6.2.3 Authentication by LDAP entries

The LDAP authentication provides access to a centralized X.500 directory where information about users, networks, and systems (actually any kind of information) is stored.

Tip: In V5R1, LDAP was included in the OS/400 base, but it still displays as option 32 for backwards compatibility with all applications. Starting with V5R2, 5722-SS1, option 32 is no longer displayed when using GO LICPGM, option 10 (Display installed licensed programs). LDAP in your iSeries is always available for any application use.

Prior to reading this section, you should:

- ▶ Be familiar with basic LDAP concepts and configuration
- ▶ Have an LDAP server already configured on your system, or have administrator access to an external LDAP server

If you have not met one or both these requirements, refer to the iSeries Information Center for documentation about your current OS/400 version and release under **Networking → TCP/IP applications, protocols, and services → Directory Server (LDAP)**. You can find the Information Center on the Web at:

<http://publib.boulder.ibm.com/pubs/html/as400/infocenter.htm>

The Information Center has detailed information about LDAP concepts and configuration and pointers to external resources. Remember that basic LDAP configuration on the iSeries is done by using iSeries Navigator. You can perform additional server configuration using the IBM SecureWay® Directory Management Tool up to OS/400 V5R2 and the IBM Tivoli® Web Administration tool in i5/OS V5R3.

We highly recommend that you refer to the IBM Redbook *Implementation and Practical Use of LDAP on the IBM @server iSeries Server*, SG24-6193.

Tip: Starting with V5R2, OS/400 heavily uses LDAP Directory Services via several OS/400 services such as Enterprise Identity Mapping (EIM), Quality of Service (QoS), and HTTP server. The base version of V5R2 shipped with an LDAP directory services version with a function set that corresponds to IBM SecureWay Directory Server V3.2. As of 16 May 2003, new PTFs were released that implement the IBM Directory Server V4.1 functionality into OS/400 V5R2. You can find the PTF numbers and detailed description of the new support on the Web at:

<http://www-1.ibm.com/servers/eserver/iseries/ldap/whatsnew41.htm>

These PTFs are not on any cumulative (CUM) package yet, so you must order them individually. Plus a new Directory Management Tool for V4.1 is available for download.

The LDAP directory server in i5/OS V5R3 is based on the IBM Directory Server V5.1.

Managing the directory

Starting with i5/OS V5R3, a new tool has been introduced to manage the LDAP directory. It is called IBM Tivoli Directory Server Web Administration Tool. This is a browser-based tool that is launched from the iSeries Tasks page. For information about the Directory Management Tool that was used in V5R1 and V5R2, see the IBM Redbook *Implementation and Practical Use of LDAP on the IBM @server iSeries Server*, SG24-6193.

Important: The IBM Tivoli Directory Server Web Administration Tool runs as a WebSphere application under the WebSphere system instance. The instance is plugged in to the HTTP Server *Admin instance. To enable the WebSphere system instance, you need to modify the General Server Configuration properties of the *Admin instance and select Yes for starting the system application server instance when the Admin server is started. For more information about starting the Web administration tool, refer to the iSeries Information Center under **Networking → TCP/IP applications, protocols, and services → Directory Server (LDAP) → Get started → Web administration** at:

<http://publib.boulder.ibm.com/infocenter/iseries/v5r3/ic2924/index.htm>

After you complete installation, start the application server system instance, and add your i5/OS system to the list of servers to be managed by the administration tool, follow these steps:

1. Open a Web browser and enter the following URL to start the iSeries Tasks page:
`http://iseries_hostname:2001`
2. Click **IBM Directory Server for iSeries**.

3. On the Tivoli Directory Server Web Administration Tool login page (Figure 6-12), select the system your LDAP server runs on and enter your directory server's credentials. The distinguished name (DN) must have the authority to manage entries in the directory. You can use the administrator DN for this task. By default, this DN is cn=admin.

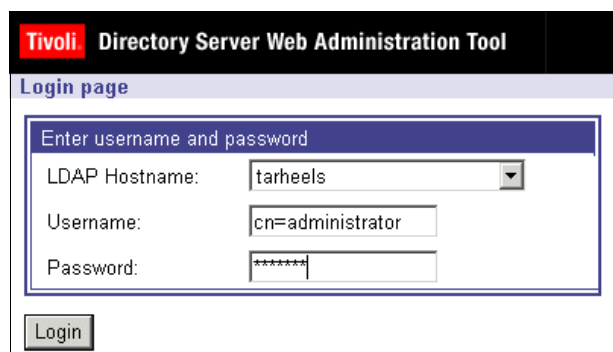


Figure 6-12 Tivoli Directory Server Web Administration Tool Login page

Note: You can only manage LDAP directory servers that have been added to the list of managed servers via the Directory Server Web Administration Tool console. By default, the console can be accessed via the user name superadmin with the password secret. We recommend that you change the password for the administrator.

Click **Login** to bind to the directory server.

4. Group users that need to access a protected resource. The IBM Directory Server V5.1 provides realms and groups to achieve this goal. From the left navigation bar, expand **Realms and templates**.
5. Click **Add user template**.
6. In the Add user template panel (Figure 6-13), you can create a user template to provide defaults for new users that are added to a realm. Enter a template name and the parent DN that stores the new entry. Click **Next** at the bottom of the page.

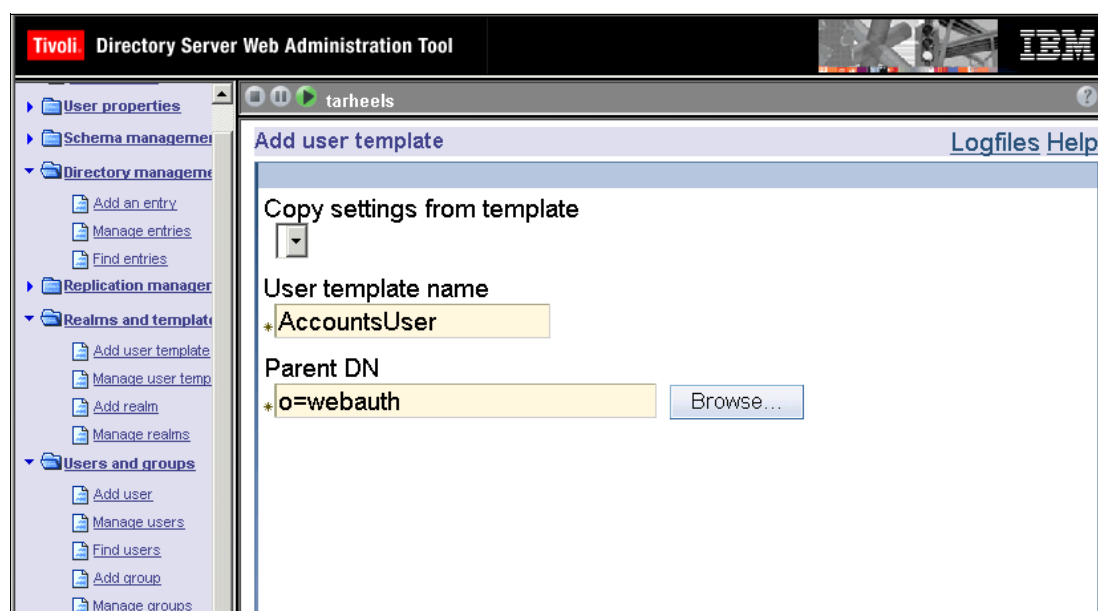


Figure 6-13 Tivoli Directory Server Web Administration Tool: Adding a user template

7. Select the object classes you want to assign to the template. In this case, only the inetOrgPerson structural object class is selected. Click **Next** to continue.
8. In the Edit tab panel, select the Tab name, which in this example is **Required**. Then click the **Edit** button.
9. Now you see the panel shown in Figure 6-14. If necessary, edit the template defaults for the Required attributes. In our example, we added the uid and userPassword attributes. Click **Finish** to create the template.

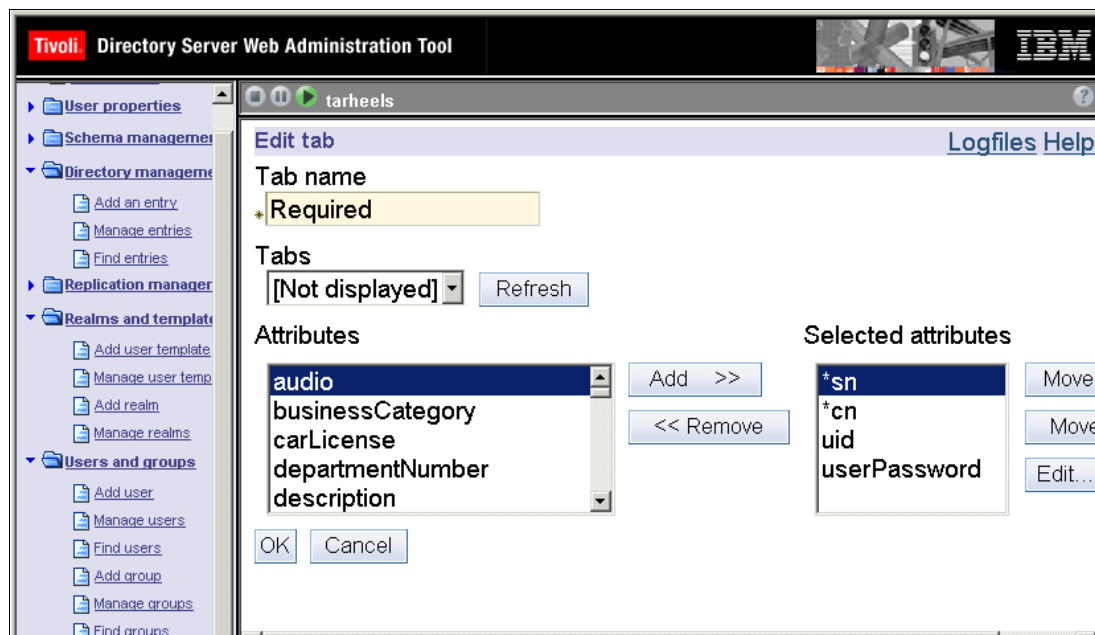


Figure 6-14 Tivoli Directory Server Web Administration Tool: Editing attributes

10. In the left navigation pane, click **Add realm** to create a new realm for the users who want to access the protected resource.
11. In the Add realm panel (Figure 6-15), enter a name and a parent DN that will store the new entry as a leaf in the directory information tree (DIT). Click **Next** at the bottom of the page.

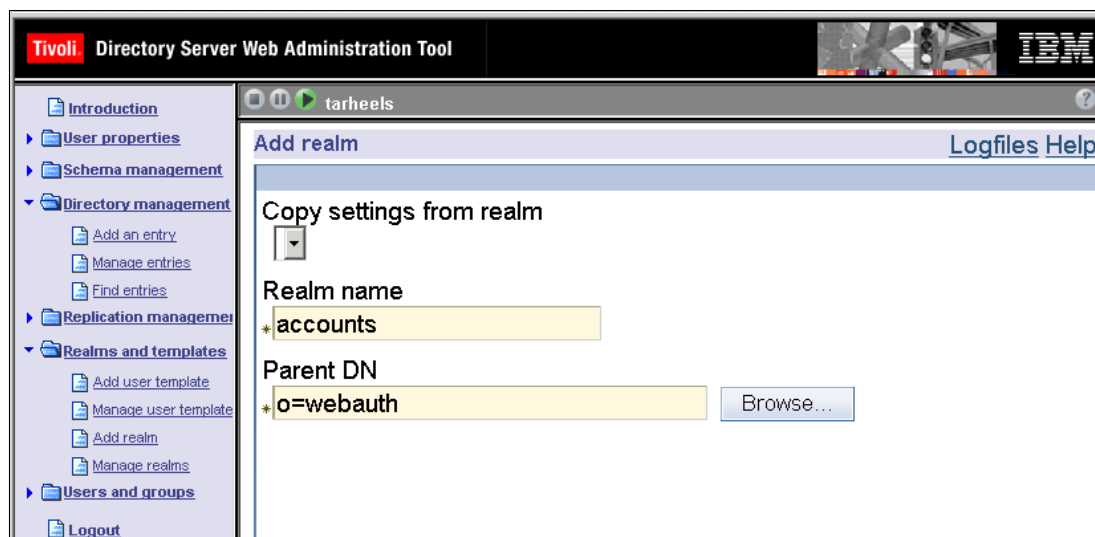


Figure 6-15 Tivoli Directory Server Web Administration Tool: Adding a realm

12. Select the user template that you just created and click **Finish** to create the realm.
13. Create a user entry to use for authentication for the protected HTTP server resource.
 - a. In the left navigation bar, expand **Users and groups**.
 - b. Click **Add user**.
 - c. In the Select the realm panel (Figure 6-16), select the realm.
 - d. Click **Next** at the bottom of the page.

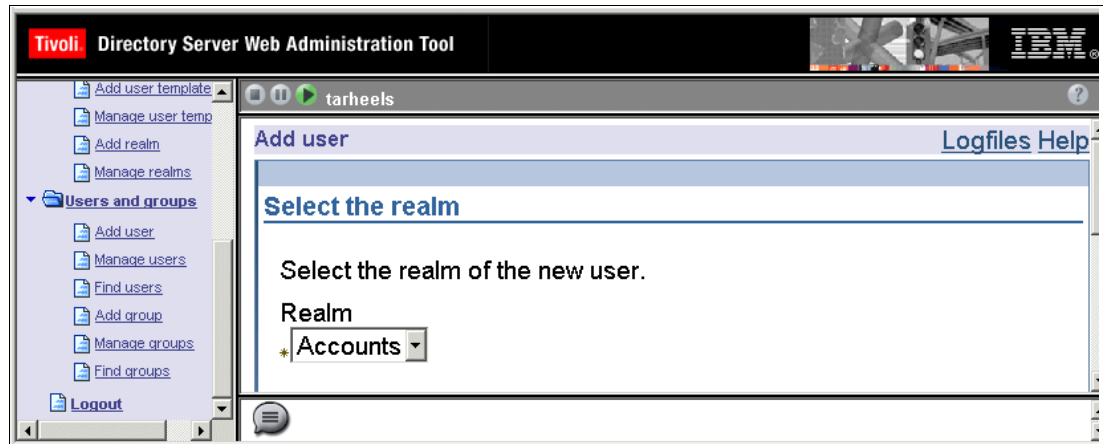


Figure 6-16 Tivoli Directory Server Web Administration Tool - Adding a user

- e. In the Naming attribute panel (Figure 6-17), fill in the attribute values. In this example, the user template asks only for the required attributes sn (last name), cn, uid (user identifier), and userPassword (password for authentication).

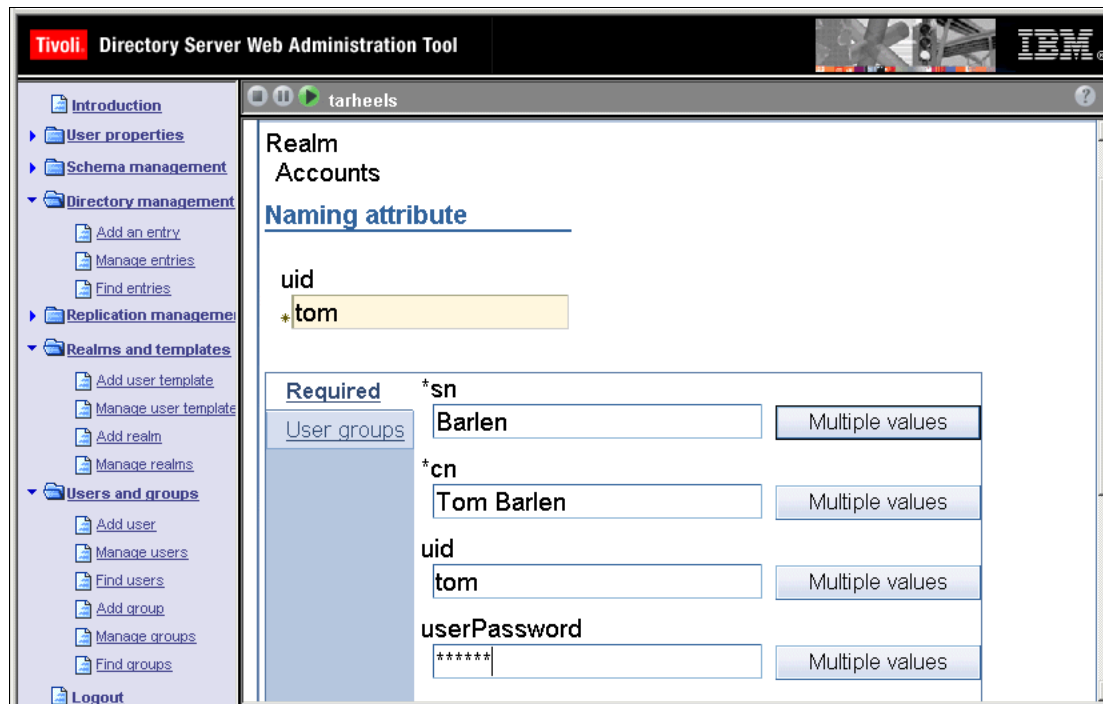


Figure 6-17 Tivoli Directory Server Web Administration Tool: Naming attributes

Important: The userPassword attribute is a special attribute in a LDAP directory. It can store the password in a protected fashion. The protection method is configured via the IBM Directory Server properties in iSeries Navigator. The Password tab contains the password encryption level and properties for the password policies.

- f. Click **Finish** to create the new user entry.

Implementation

Now that the LDAP server is configured and contains a user entry, you are ready to configure your HTTP Server (powered by Apache) to use basic authentication and LDAP:

1. From the Server area list, select the context that you want to protect.
2. In the left pane, under Tasks and Wizards, select **LDAP Configuration**.
3. In the LDAP configuration panel (Figure 6-18), enter the directory path to your LDAP configuration file. The file name does not have to exist, but you need to specify an existing path when you create a new file. Click **Next**.

Tip: To avoid path errors, browse for the file and let the system enter the name for you.

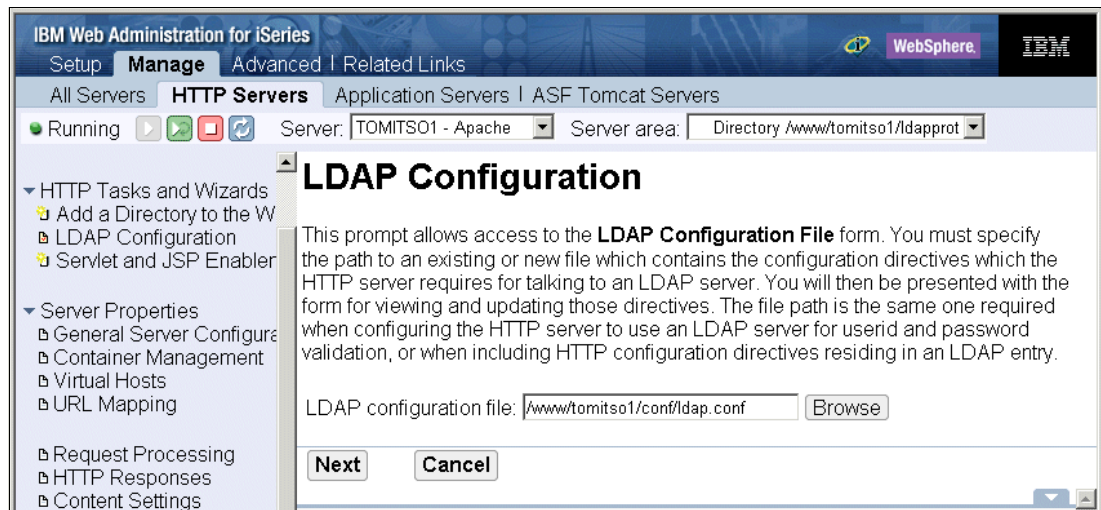


Figure 6-18 Basic Authentication with LDAP: LDAP configuration file

4. In the LDAP Configuration File panel (Figure 6-19), select the **General Settings** tab.
5. On the General Settings page, complete these tasks:
 - a. Enter your LDAP server description.
 - b. Under LDAP server location, enter the host name or IP address, the port number, and the search base DN. The search base DN refers to the position in the DIT where you store the Web users.
 - c. Scroll down and select **Basic Authentication(DN and password)**.

- d. Enter your Server DN and server password.

The HTTP server uses this DN to bind to the LDAP directory server. The DN must have the authority to search and read the directory portion that holds the user entries. This includes read authority of the userPassword attribute. Even though most users enter the directory administrator (cn=admin) as the DN, it could be any other DN as long as it has the proper permissions.

- e. Click **Apply** and then click **OK** to save the general settings.

Depending on the object classes you used for your user entries, you may need to modify the search filter on the User Authentication tab. By default, the search filter is:

```
(&(objectclass=person)(|(cn=%v1* %v2*)(uid=%v1)))
```

This filter causes the HTTP server to search for entries that belong to the person object class and where the value entered at the login prompt matches a common name or uid in the specified search base. The specified object class can also be an inherited class.

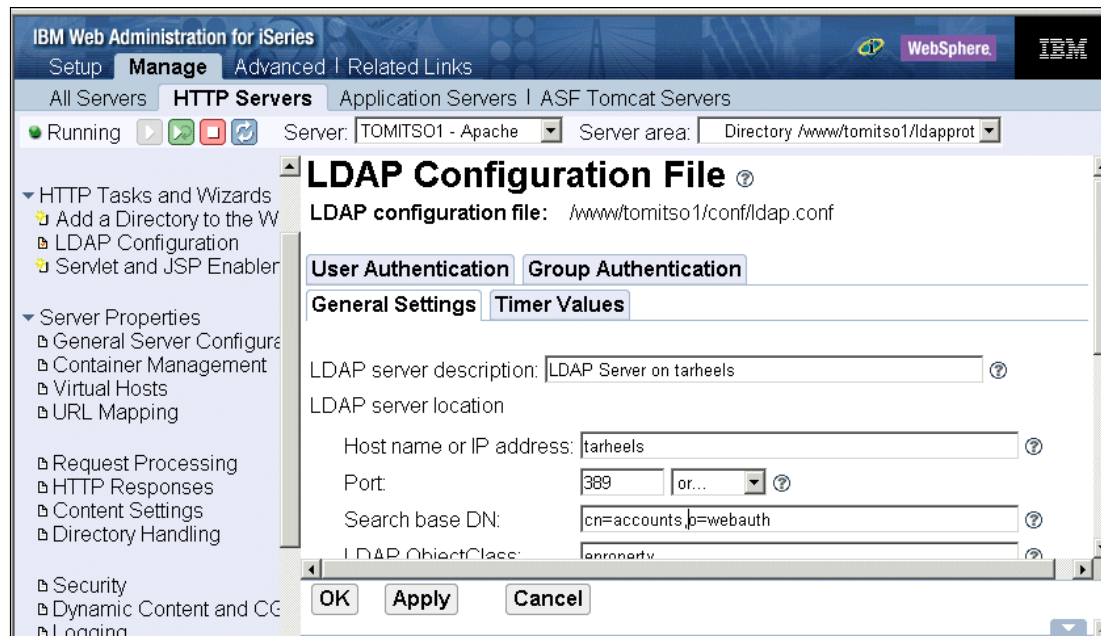


Figure 6-19 Basic Authentication with LDAP: LDAP server description

6. In the left pane under Server Properties, select **Security**.
7. Select the **Authentication** tab.
8. On the Authentication page (Figure 6-20), complete these tasks:
 - a. Select **Use user entries in LDAP server**.
 - b. Enter a name for the realm.
 - c. Enter the LDAP configuration file.
 - d. Scroll down and specify a user profile to process requests.

Specify a user name to process requests. In a case where you are using LDAP directory entries to verify the identity of the remote user, the authenticated user has no connection to an OS/400 user profile. Therefore, you cannot process the request under the client authority. However, you can enter a user profile name for the OS/400 user profile to process requests. This user must have proper authority to the protected resources. If this parameter is left blank, the default server profile QTMHHTTP is used to access static content and QTMHHTTP1 for CGI programs.

- e. Click **Apply** to save the settings.

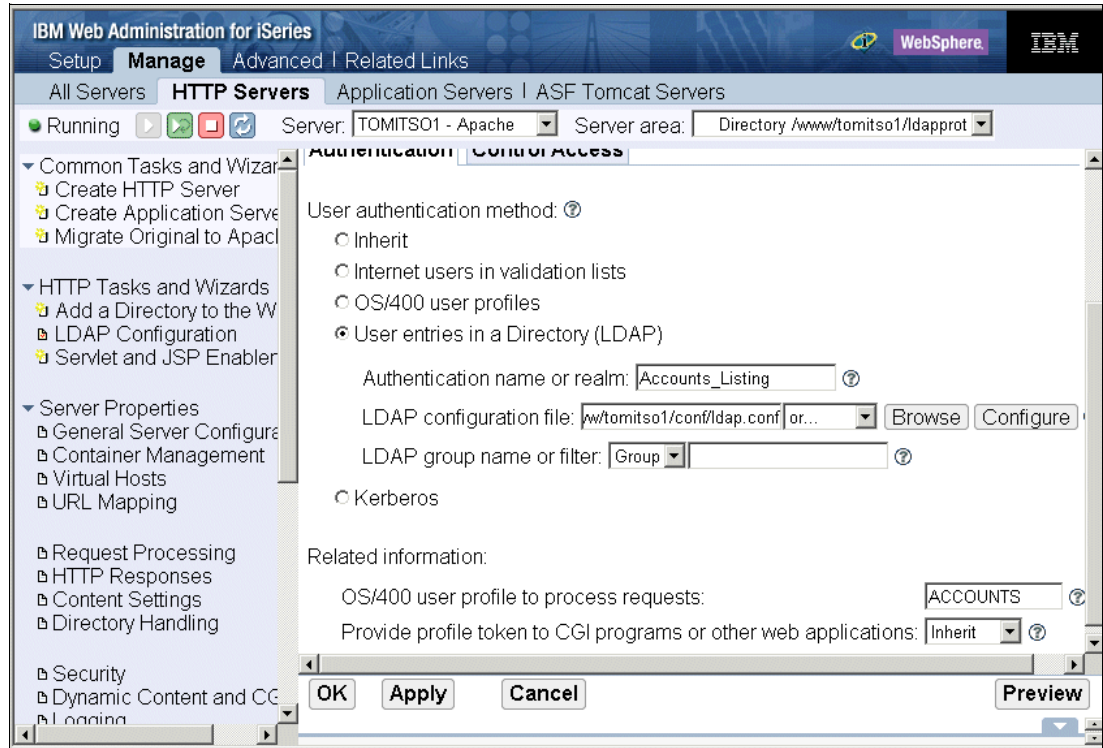


Figure 6-20 Basic Authentication with LDAP: User entry selection

9. Click the **Control Access** tab.
10. On the Control Access page, complete these tasks:
 - a. Select **All authenticated users (valid user name and password)**.
 - b. Click **OK** to save the changes and close the Security window.
11. Restart your server instance. Point your Web browser to the context you just protected. You are then prompted for a user name and password. Use the one that you entered into the LDAP directory.

6.3 Authenticating users via Kerberos

Kerberos is a network authentication protocol that was developed by the Massachusetts Institute of Technology (MIT) as part of their Athena project. It was designed to provide strong authentication for client/server applications by using secret-key cryptography. Kerberos is a ticket-based authentication system that provides an alternative to user/password or X.509 certificate authentication. Since Kerberos uses additional ports for its IP services, such as the Key Distribution Center (KDC), it is not always feasible to use it for authentication in an Internet environment. It is rather useful to provide single signon (SSO) capabilities for an intranet environment.

For more information about the Kerberos authentication protocol, refer to the MIT Web site at <http://web.mit.edu/kerberos/>

Important: You can only use Kerberos authentication for resources that are protected by the HTTP Server (powered by Apache). Proxy authentication with Kerberos is not supported due to limitations in Web browsers.

With the HTTP Server (powered by Apache), you can use Kerberos on its own or in conjunction with EIM to authenticate Web users to the Web server as illustrated in Figure 6-21.

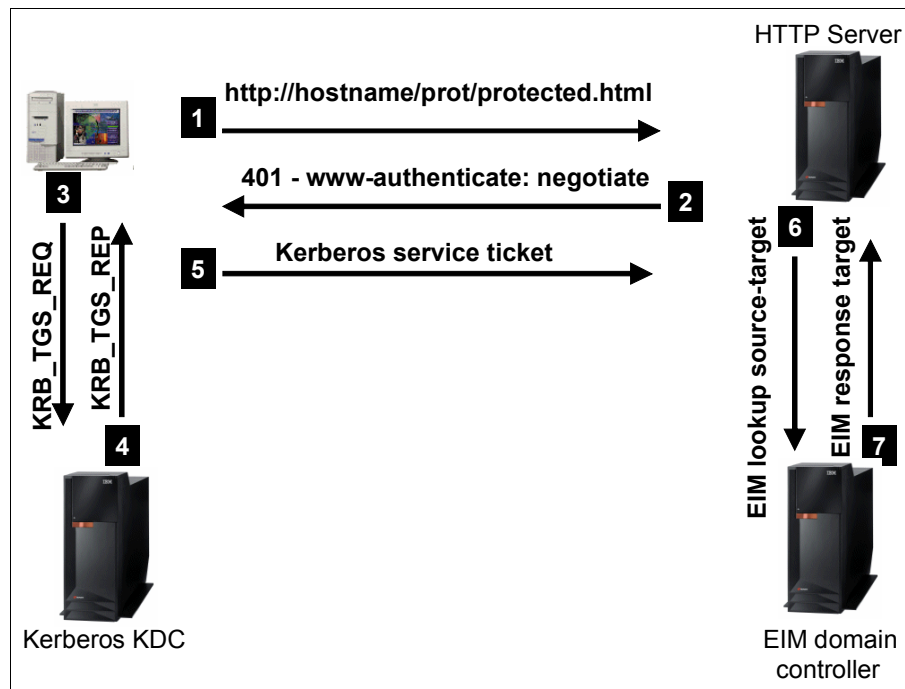


Figure 6-21 HTTP Server (powered by Apache) Kerberos authentication

The Kerberos authentication process in Figure 6-21 follows this flow:

1. The browser accesses a URL that is protected by the Kerberos authentication mechanism.
2. The HTTP server returns a 401 (authorization required) response, causing the browser to obtain credentials. Opposed to basic authentication where the authentication header contains the value `basic`, the HTTP header now contains the `www-authenticate: negotiate` value. The negotiate option is used for Kerberos authentication.
3. Assuming that the workstation is already successfully authenticated to the KDC, the browser application requests a service ticket for the HTTP server from the KDC. The service principal for the HTTP server is `HTTP/hostname@KERBEROS_REALM`.
4. The KDC issues a service ticket and returns it in the Kerberos TGS reply to the workstation.
5. The browser sends the Kerberos service ticket embedded in the HTTP request to the HTTP server. The HTTP server validates the service ticket. If you configured the HTTP server to process the client request under a specific user profile (not under the client profile), the authentication is complete.
6. If you selected the configuration option to process the client request under the client user profile, the authentication process continues with this step. Since the user principal in the service ticket does not relate to an OS/400 user profile, the HTTP server performs an EIM lookup operation to the EIM domain controller. Specifically, the HTTP server asks for a target association in the OS/400 user registry for the given Kerberos user principal in the Kerberos user registry.
7. The EIM domain controller looks up and returns the target association, which corresponds to an OS/400 user profile, to the HTTP server. The HTTP server, if configured that way,

switches to the target OS/400 user profile and performs resource access under this user profile.

Note: The workstation user is not prompted for any user input during the authentication process. Kerberos authentication is totally transparent to the user.

6.3.1 Getting ready for Kerberos authentication

Using Kerberos in combination with EIM for HTTP server authentication requires some prerequisites. The configuration steps in 6.3.2, “Implementing Kerberos Web authentication” on page 122 assume that:

- ▶ Kerberos and EIM support availability on the iSeries server are as follows:
 - HTTP Server for iSeries (powered by Apache) V5R2 with group PTF SF99098 level 13 (December 2003)
 - HTTP Server for iSeries (powered by Apache) V5R3
- ▶ The client operating systems must support Kerberos authentication. The list of supported clients includes:
 - Windows 2000 Professional
 - Windows XP
 - Linux clients
- ▶ At the time this redbook was written, the following Web browsers supported Kerberos authentication:
 - Microsoft Internet Explorer 5.5 and later
 - Mozilla 1.7 for UNIX or Linux
 - Mozilla 1.8 and Firefox 1.0 for Windows (still an Alpha version at the time of writing this book)
- ▶ The configuration presented in this chapter assumes that a Kerberos environment already exists and that the iSeries server and the workstation are already configured to be members of that Kerberos realm.
- ▶ If you want to access protected resources under the client user profile (user profile value %%CLIENT%%), you also need to set up an EIM domain. For this implementation, the EIM domain controller is already set up.

Note: For an excellent source of information about setting up SSO with Kerberos and EIM in a Windows and iSeries environment, see *Windows-based Single Signon and the EIM Framework on the IBM @server iSeries Server*, SG24-6975.

6.3.2 Implementing Kerberos Web authentication

The following steps guide you through the setup of the HTTP server for Kerberos authentication. The example also shows the necessary steps to enable Kerberos authentication for Microsoft Internet Explorer 6. The examples assume that the Kerberos realm name is *ISERIES.IBM.COM* and the iSeries host name is *tarheels.barlen.net*.

Preparing the Kerberos environment

Perform the following steps to add the Kerberos service principal for HTTP Kerberos authentication.

1. If have not already configured your OS/400 Kerberos environment for the HTTP service, start iSeries Navigator and connect to the system that will host your HTTP server instance.
2. Expand **Security** → **Network Authentication Service** and click **Configure Network Authentication Service** from the list of Security Tasks in the lower pane of the iSeries Navigator window.
3. The Network Authentication Service (NAS) setup wizard starts. Enter the correct values for your Kerberos realm, the KDC, and the password server. If your KDC is a Windows domain controller, select the **Microsoft Active Directory is used for Kerberos authentication** option.
4. Continue with the wizard. Select **HTTP Server powered by Apache** and click **Next**.

Note: If you do not see the HTTP server option in the wizard, you may experience one of the following problems:

- ▶ You have an older version of iSeries Navigator installed. You need to upgrade iSeries Navigator to see this option.
- ▶ You are connected to a V5R2 system that is not aware of HTTP Server Kerberos support.

Alternatively, you can manually add the keytab entry by starting QShell from a 5250 session and enter the following commands:

```
keytab add -p password HTTP/tarheels.barlen.net@ISERIES.IBM.COM
keytab add -p password HTTP/TARHEELS.BARLEN.NET@ISERIES.IBM.COM
```

5. In the Create HTTP Keytab Entry window (Figure 6-22), enter a password for the HTTP service principal keytab entry. Click **Next**.

The screenshot shows a window titled "Network Authentication Service Configuration - Create HTTP Keytab Entry". It contains the following elements:

- An icon of a padlock and a key.
- Text: "The HTTP server may use Kerberos to authenticate clients in a single signon environment. To enable this, keytab entries must be defined for the HTTP service." and "What password will be used for the service principals? The password used when creating the keytab entries and defining the principal on the KDC must be the same."
- A text box labeled "Keytab:" containing the path "UserData/OS400/NetworkAuthentication/keytab/krb5.keytab".
- A list box labeled "HTTP Principals" containing two entries: "HTTP/tarheels.barlen.net@ISERIES.IBM.COM" and "HTTP/TARHEELS.BARLEN.NET@ISERIES.IBM.COM".
- Two password input fields labeled "Password:" and "Confirm password:", both showing masked characters (asterisks).
- Navigation buttons at the bottom: "< Back", "Next >", "Finish", and "Cancel".

Figure 6-22 Network Authentication Service wizard: HTTP keytab entry

Depending on the selected wizard options, one or two keytab entries are added for the HTTP service.

6. Complete the wizard. The wizard adds a keytab entry to the keytab file /QIBM/UserData/OS400/NetworkAuthentication/keytab/krb5.keytab. The entries in this file are used by applications that are processed with Kerberos during Kerberos authentication.

The OS/400 preparation for HTTP Kerberos authentication is completed. In the remaining steps, you need to register the HTTP service principal to the KDC database. An outline of the configuration steps are included for a Windows 2000 domain controller and the i5/OS KDC.

Adding a service account to Windows Active Directory

The following steps assume that your Windows domain controller is the KDC for your network. They also assume that you selected the NAS wizard option that Microsoft Active Directory is used for Kerberos authentication.

1. Copy the batch file that the NAS wizard created to your Windows domain controller.
2. In the Windows domain controller, start a command prompt and run the batch file.

The batch file creates a service account and maps the HTTP service principal to the new service account. If you selected the NAS wizard option to not store the password in the batch file, you are prompted on the Windows domain controller to enter a password. This password must match the password that you entered in the NAS wizard for the HTTP service keytab entry.

Note: Two commands in the batch file, by default, are not installed on a Windows server. One command is the **ktpass** command. This command is part of the Windows Support Tools and must be installed separately from the Windows installation CD. The second command is **setspn**, which is part of the Windows 2000 Server Resource Kit. It can also be downloaded from the Microsoft Web site.

Adding service accounts to the i5/OS KDC

If you use the i5/OS KDC as your primary Kerberos authentication service in your network, perform the following steps to add the HTTP service principals to the KDC database.

1. Within a 5250 command prompt on the system that runs the KDC, start the OS/400 Portable Application Solutions Environment (OS/400 PASE) shell with the command:

```
CALL QP2TERM
```

2. Start the KDC administration interface with the command:

```
/usr/krb5/sbin/kadmin.local
```

3. Add the service principals with the following **kadmin** commands:

```
addprinc -pw password HTTP/tarheels.barlen.net@ISERIES.IBM.COM  
addprinc -pw password HTTP/TARHEELS.BARLEN.NET@ISERIES.IBM.COM
```

The password must match the password that you entered for the HTTP service keytab entry in the NAS wizard.

Setting up the HTTP server for Kerberos authentication

Use the following setup to configure the HTTP server to protect a resource and authenticate the users via Kerberos.

1. From the Server area list, select the context that you will protect.
2. In the left pane, under Server Properties, select **Security**.
3. In the right panel, select the **Authentication** tab.

4. On the Authentication page, complete these tasks:
 - a. Select **Kerberos**.
 - b. Select the client authority that you will use to perform access to this resource. The available options are:
 - **Enabled**: When accessing the resource, the server temporarily switches to the user profile of the authenticated user and performs access under this user profile. A special value %%CLIENT%% is used on the UserID directive. The UserID directive overrides the ServerUserID directive. This option uses EIM to provide the source to target identity mapping.
 - **Disabled**: Access to the protected resources is, by default, performed under the QTMHHTTP profile for static pages and QTMHHTTP1 profile for CGI programs unless the ServerUserID directive specifically names another user profile. Using this option, EIM is not used in the authentication phase. Only the Kerberos service ticket is needed to complete the authentication. Access to the protected resource is performed under the hardcoded user profile.
 - c. If Disabled is selected for the client authority, enter the profile name that the server will use to access this resource. You have to enter a user profile name. No special values are allowed.
 - d. Click **Apply** to save your settings.

Tip: When dealing with OS/400 user profiles, always remember that users with *ALLOBJ authority are not subject to any access restriction on the file system, not even an explicit *EXCLUDE.

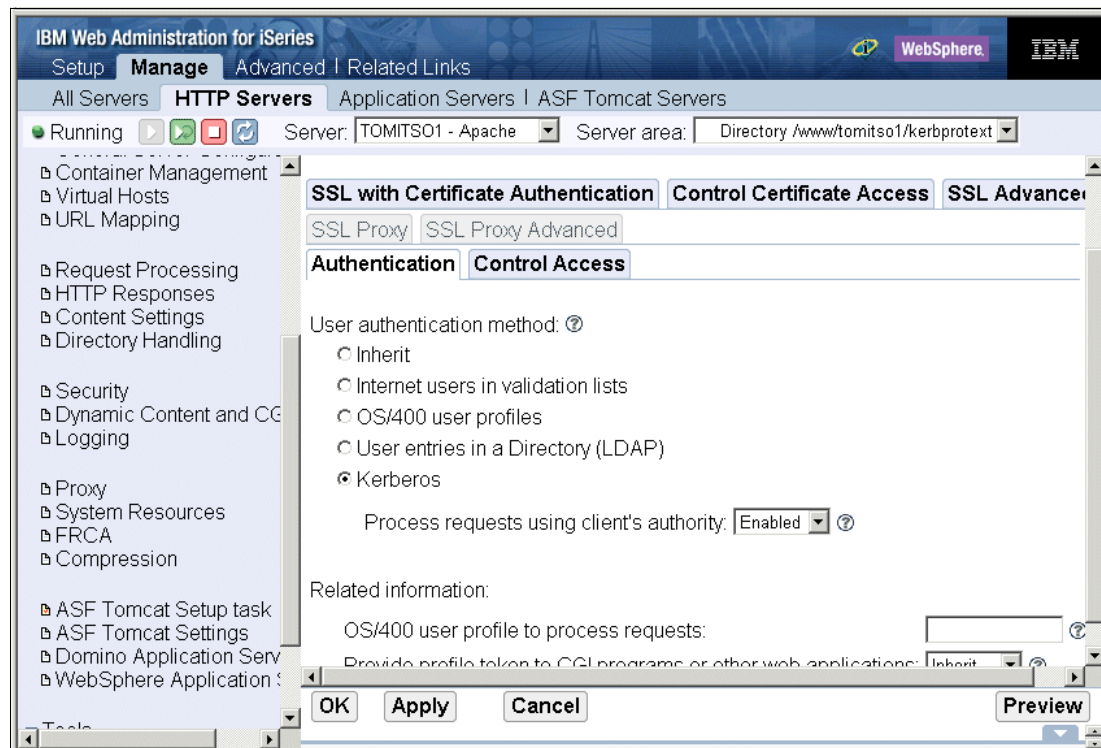


Figure 6-23 Authentication with Kerberos

5. Click the **Control Access** tab.

6. On the Control Access page, follow these steps:
 - a. Select **All authenticated users (valid user name and password)**.
 - b. Click **OK** to save the changes and close the Security window.
7. Restart your server instance.

Enabling Kerberos authentication for the Microsoft Internet Explorer

By default, Kerberos authentication is turned off for the Internet Explorer browser. The following steps enable Kerberos authentication for the browser.

1. Start Internet Explorer.
2. In the browser action bar, click **Tools** and then **Internet Options....**
3. In the Internet Options window (Figure 6-24), complete these tasks:
 - a. Click the **Advanced** tab.
 - b. Scroll down to the security section and select the **Enable Integrated Windows Authentication (requires restart)** option.
 - c. Click **OK** to save the changes and restart your browser.

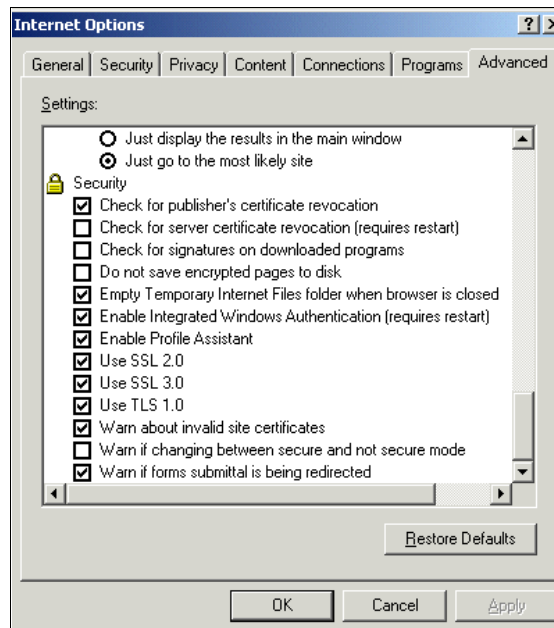


Figure 6-24 Internet Explorer - Internet Options

To verify Kerberos authentication with your browser, you must log into a Kerberos realm. If you selected the option to perform access to the protected resource under the client's authority, you must also have an EIM identifier with the corresponding source and target associations defined in the EIM domain controller.

Point your browser to the protected resource. You can open the HTTP server access log to see which user principal was used to authenticate to the server. Example 6-1 shows an access log authentication message.

Example 6-1 HTTP server access log

```
172.25.10.142 - - [09/Sep/2004:15:31:33 +0200] "GET /kerbprot/ HTTP/1.1" 401 205
172.25.10.142 - thomas@ISERIES.IBM.COM [09/Sep/2004:15:31:34 +0200] "GET /kerbprot/
HTTP/1.1" 200 876
```

Important: The host name that you enter as part of the URL to access the protected resource must be the name that you registered as part of the service principal in the KDC and the i5/OS keytab file. For example, if you enter `http://sysa/protect`, but you registered the service principal `HTTP/sysa.iseries.com@REALM.COM`, the URL host name `sysa` does not match the service principal name `sysa.iseries.com`. In this case, the KDC does not find a corresponding entry. To enable users to use different host names in the URL, such as short names and fully qualified names, add multiple service principals to the KDC and the keytab file.

6.4 Encrypting your data with SSL and TLS

This section is written based on the assumption that you are already familiar with digital certificates and the Digital Certificate Manager (DCM) interface. Also, a valid server certificate must already be installed or created on the system. Two good IBM Redbooks to help you in this area are:

- ▶ *IBM @server iSeries Wired Network Security: OS/400 V5R1 DCM and Cryptographic Enhancements*, SG24-6168, offers step-by-step instructions on digital certificate creation and management.
- ▶ *AS/400 Internet Security: Developing a Digital Certificate Infrastructure*, SG24-5659, documents information about downloading and maintenance of digital certificates from VeriSign. Appendix C in this book guides you through obtaining a trial certificate from VeriSign. You can also go directly to the VeriSign link at:

<http://www.verisign.com/client/index.html>

In addition, the HTTP Server (powered by Apache) uses module `mod_ibm_ssl` for all the authentication and encryption for SSL and Transport Layer Security (TLS). You can find the manual for this module on the Web at:

http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzaie/rzaiemod_ibm_ssl.htm

6.4.1 Enabling SSL

As an overview of the operations necessary to configure SSL/TLS with your HTTP Server (powered by Apache), Table 6-2 lists a series of goals. The first three are handled by the administration GUI of the HTTP Server (powered by Apache) and result in new directives in your configuration file. The rest depends on the configuration steps necessary with the DCM.

Tip: Since you need to add HTTPS as a new protocol for your server, you must create a virtual host context to manage your SSL-secured communications.

Table 6-2 Enabling SSL

Goal	GUI configuration steps	Apache final configuration file
Add a new port for SSL-secured communications. The well-known port for SSL is 443. In this example, we use port 44306.	In your server's General Server Configuration panel, click the General Settings tab and add the new port number.	Listen 44306
Create a virtual host context that contains the SSL directives.	In the Virtual Hosts panel, add a Virtual Host context listening on the new port.	33 <VirtualHost *:44306>
Enable SSL for the virtual host.	Select the new virtual host as the active context. Select Security and then click the SSL with Certificate Authentication tab. Under SSL, click Enable SSL .	2 LoadModule ibm_ssl_module /QSYS.LIB/ QHTTPSVR.LIB/QZSRVSSL.SRVPGM ... 35 SSLAppName QIBM_HTTP_SERVER_TOMITSO1 36 SSLEnable 37 SSLCacheDisable 38 </VirtualHost>
Assign a digital certificate to the server.	In the DCM GUI, open the *SYSTEM certificate store. Under Fast Path, select Work with server applications . From the list on the right, select your server. Assign a valid certificate and make sure that the Certificate Authority (CA) is marked as trusted in the CA Trust List. No change is made to the HTTP configuration file.	
Install the local CA on the client PC.	Select Install Local CA Certificate on Your PC .	
Restart the server and test your SSL configuration.	Point your browser to: https://servername:SSLport Remember that you can access the virtual host only through the HTTPS protocol now.	

Implementation

You can enable a new port, create a virtual host context, and tell the server to load the SSL module used for secure communications. Follow these steps:

1. From the Server list, select your server name.
2. From the Server area list, select **Global Configuration**.
3. In the left pane, under Server Properties, select **General Server Configuration**.
4. Click the **General Settings** tab.

5. On the General Settings page (Figure 6-25), complete these steps:
 - a. Click **Add** to enable the new port we will use for SSL communications. In our example, we listen on a new port of 44306 on all IP addresses on our iSeries server.
 - b. Click **Continue** to add the new port to the list of ports the server listens on.
 - c. Click **OK** to close the General Server Configuration window.

Tip: There are three options you can click to update the configuration with a new listen port. The first is to click Continue. The second is to click OK. The third is to click Apply. Although all three options update the configuration, Apply is the only option that results in a message being displayed in the lower blue message area stating that the configuration was successfully updated.

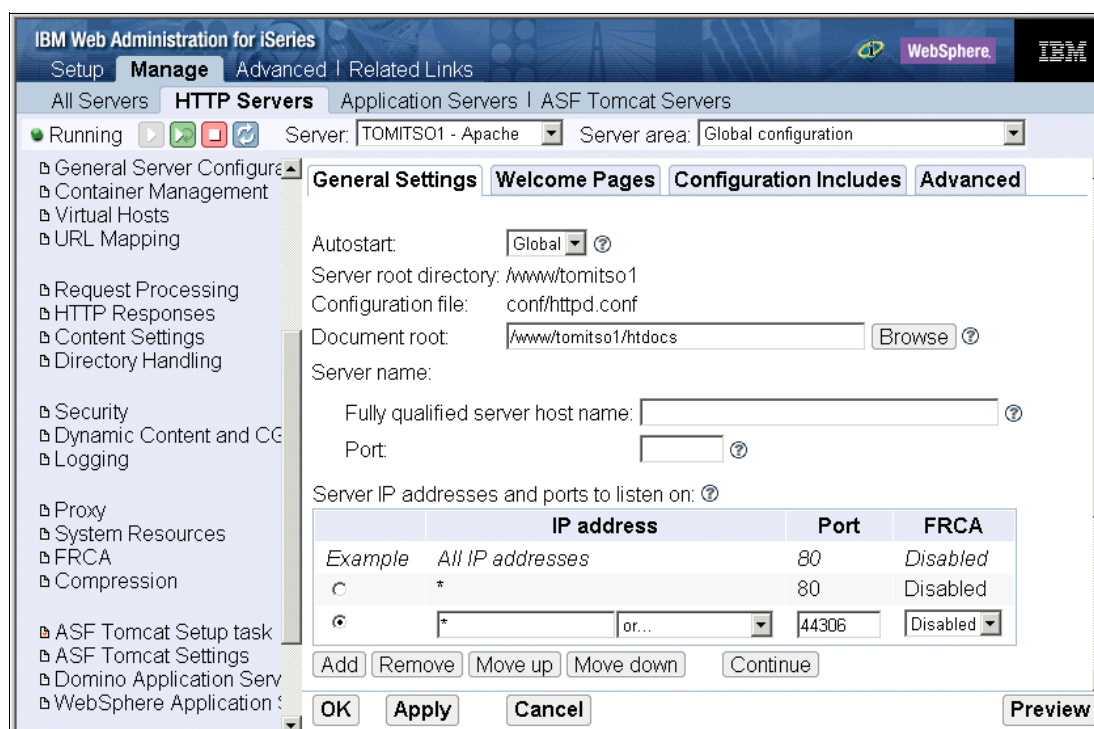


Figure 6-25 A context for SSL3

6. In the left pane, select **Container Management**.
7. In the right panel, click the **Virtual Hosts** tab.

8. On the Virtual Hosts page (Figure 6-26), follow these steps:
 - a. Scroll down to locate the Virtual host containers table.
 - b. Click **Add**.
 - c. Fill the blank fields with the IP address and port to use with SSL.
 - d. Click **OK**.

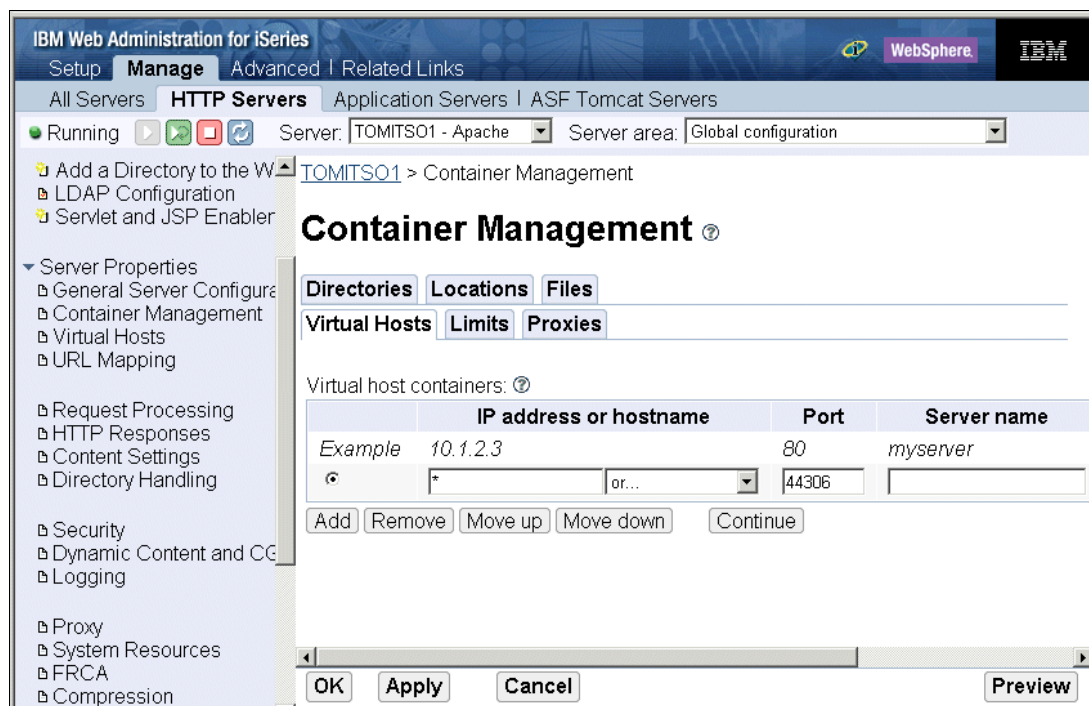


Figure 6-26 Adding an IP-based virtual host

9. To enable your server to support SSL, from the Server area, select first the virtual host context.

Important: If you do not switch to the virtual host context before continuing with the next steps, you can only access the entire server instance with SSL (HTTPS). Normal HTTP connections will not work anymore.

10. In the left pane, under Server Properties, click **Security**.
11. In the Security panel, click the **SSL with Certificate Authentication** tab.

12. On the SSL with Certification Authentication page (Figure 6-27), complete these steps:
- For SSL, select **Enable SSL**.
 - Notice the server certificate application name that is used for this server. You need this application name in the DCM configuration later in this chapter.
 - Click **Apply**.
 - Click **OK**.

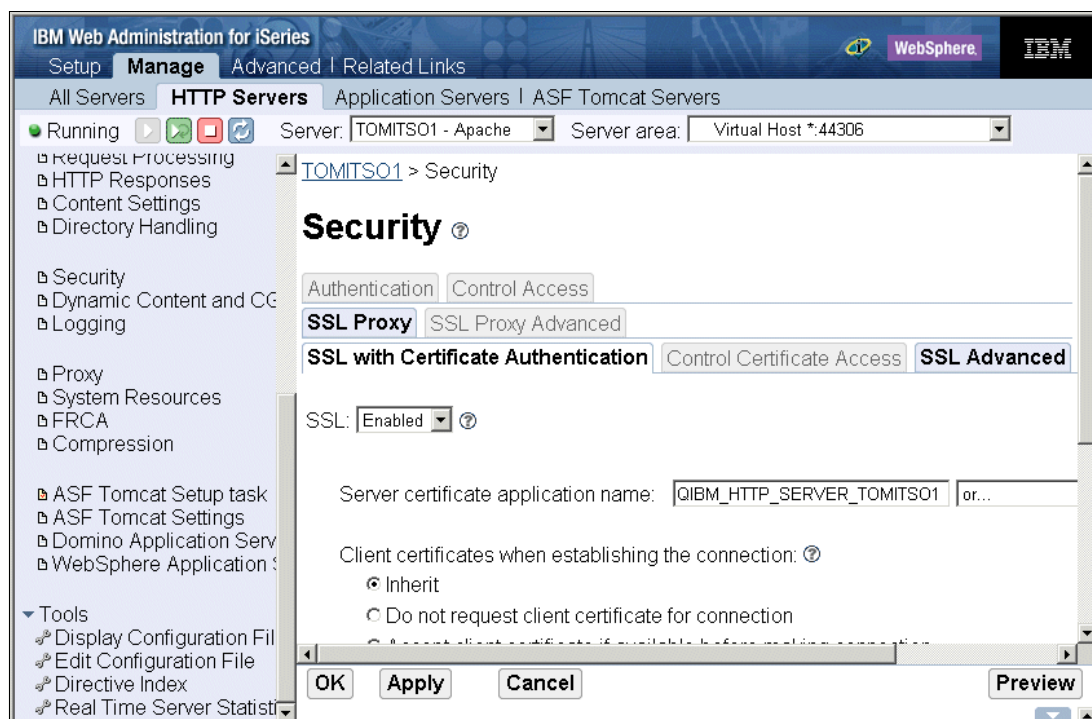


Figure 6-27 Enabling SSL

At this point, the GUI adds all the proper LoadModule and SSL directives into your configuration file. You may want to look at all the new directives using the Display Configuration File form. In addition, the GUI registers your Application name with the DCM as a new server application.

Tip: You can also enter the following CL command to register an HTTP instance within the DCM environment:

```
CALL QHTTSPVR/QZHAPREG PARM('RegisterAppName' 'QIBM_HTTP_SERVER_name')
```

Your HTTP Server (powered by Apache) configuration is now complete and supports SSL and TLS. Next, you associate the application name (specified in Figure 6-27) with a valid server digital certificate. This is done through the DCM GUI. The following steps take you into the Digital Certificate Manager GUI.

1. From the tabs at the top of the IBM Web Administration for iSeries page, select **Related Links**.
2. You see all of the links as shown in Figure 6-28. Click **Digital Certificate Manager**.

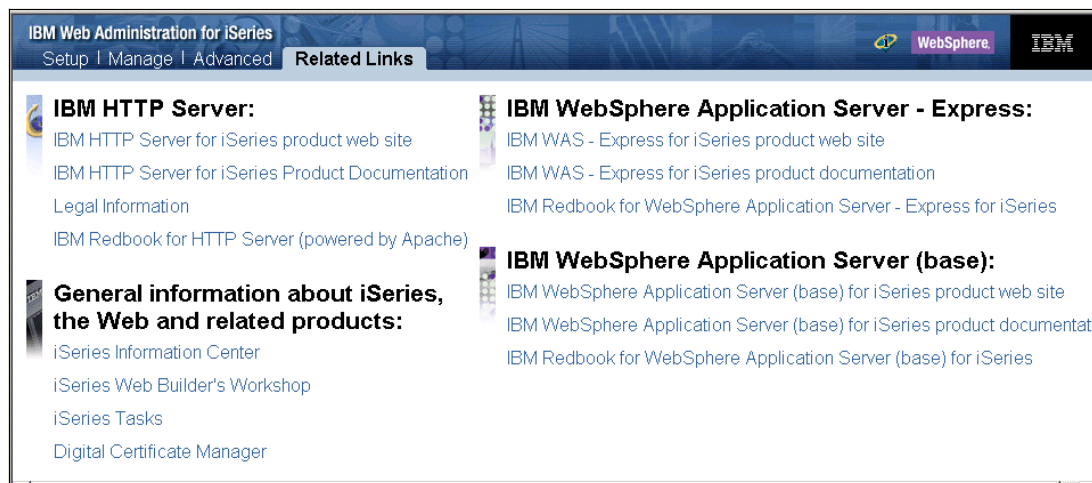


Figure 6-28 Accessing DCM

3. In the DCM main menu, complete these steps:
 - a. Click **Select a Certificate Store**. You must be signed on to the iSeries Tasks page with a user that has at least *SECADM and *ALLOBJ special authorities to see the Select a Certificate Store button.
 - b. Under Select the certificate store that you want to open, select ***SYSTEM**.
 - c. Click **Continue**.
4. Enter the Certificate Store password as shown in Figure 6-29. Enter your certificate store password and click **Continue**.

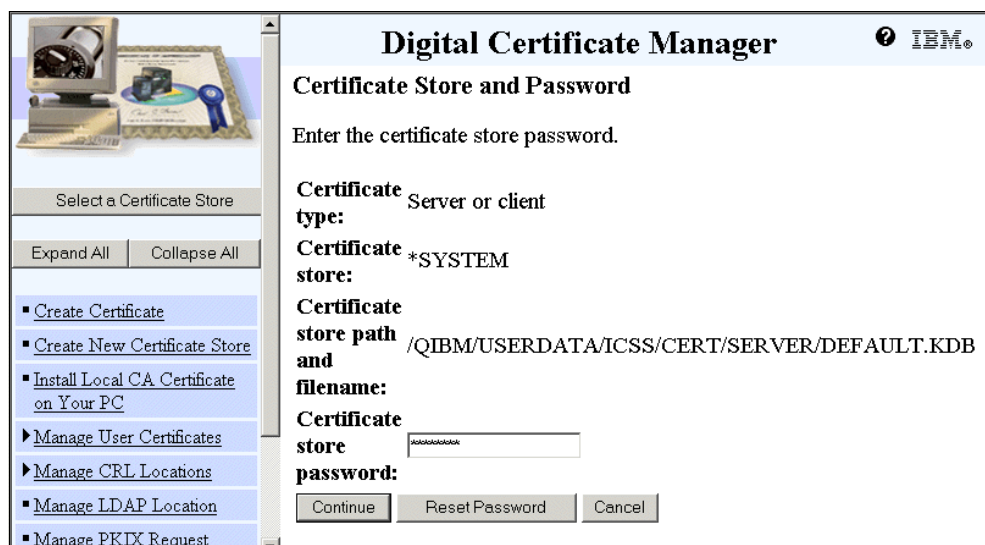


Figure 6-29 Certificate Store and Password

- After the certificate store opens, in the left navigation pane, click **Fast Path** to expand the list of options. Then select **Work with server applications**.
- On the right side, select your server application name from the list. This is the same name that you specified in Figure 6-27 on page 131. You may have to scroll down to see the name of the server instance. Click the **Work with Application** button.

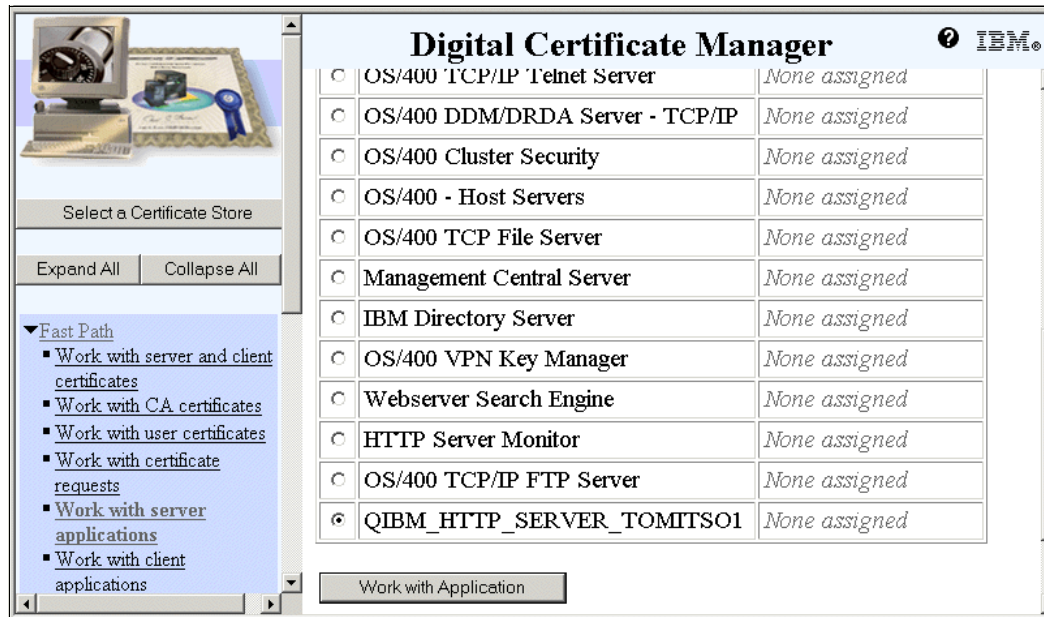


Figure 6-30 Working with server applications

The following steps help to establish the association between the server digital certificate and your HTTP Server (powered by Apache).

- Click the **Update Certificate Assignment** button.
- Select the server certificate that your application will use.
- Click the **Assign New Certificate** button.

4. You see confirmation message in the top part of the page as shown in Figure 6-31. Click **Cancel** to return to the Work with Application window.

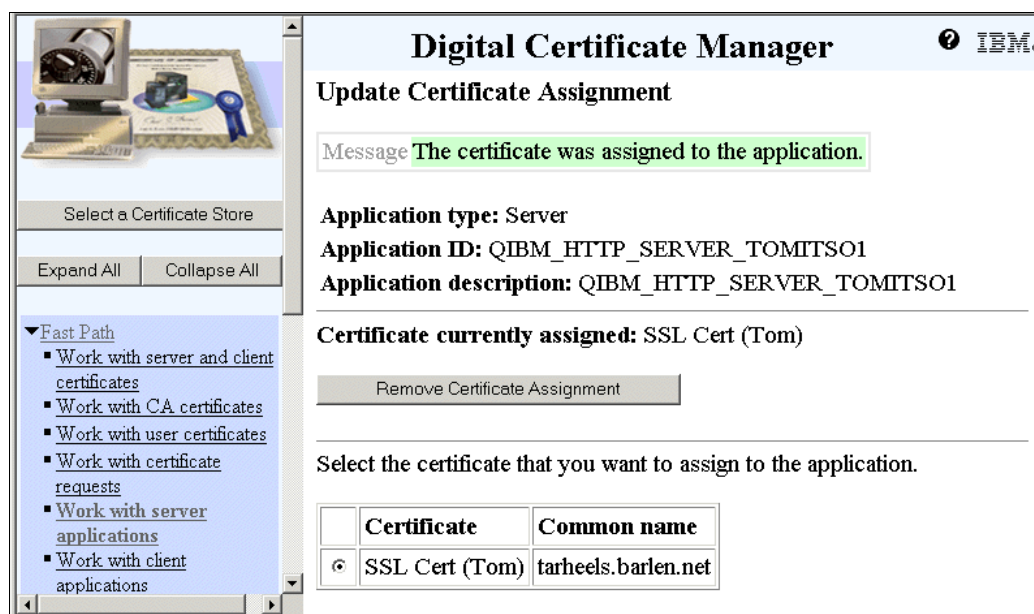


Figure 6-31 DCM Update Certificate Assignment panel

5. Check that the issuer of your certificate is listed among the trusted certificate authorities.
6. Click the **Define CA Trust List** button to add one or more CAs to the list. Remember that, in case of a local CA, you also have to install the CA certificate in your Web browser.
7. In the Define CA Trust List window (Figure 6-32), select the CAs that you want to trust and click **OK**.

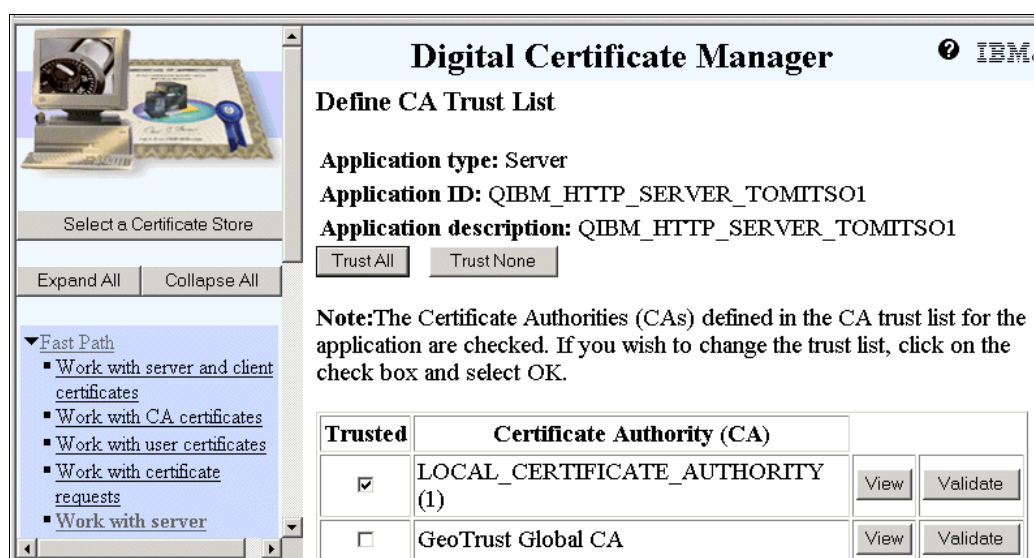


Figure 6-32 DCM Define CA Trust List panel

8. Click **Cancel** to return to the Work with Server Applications panel (Figure 6-33).



Figure 6-33 DCM Work with Server Application panel

9. Restart your HTTP server and point your browser to:

`https://servername:SSLport`

Tip: Do not forget the “s” as part of the HTTPS protocol.

When you open a SSL connection, you may see a browser window that asks you to verify the information contained inside the digital certificate, and whether you want to accept it. This window typically opens when the host name entered in the URL does not match the host name in the common name attribute of the certificate. Another reason why the window opens is because you have not installed the CA certificate of the CA that issued the server certificate in the browser certificate store. Click the lock displayed in your browser's status bar for detailed information about the secure transaction.

6.4.2 TLS upgrade

A feature that was introduced with V5R2 of the HTTP Server (powered by Apache) is the ability to allow clients to request an upgrade to TLS encryption on an unencrypted port. This allows new applications to need only one port for both normal and SSL traffic. Right now the primary user is Internet Print Protocol (IPP) client as illustrated in Figure 6-34.

Tip: Refer to Request for Comments (RFC) 2817: Upgrading to TLS Within HTTP/1.1.

Allows a client to request an upgrade to TLS encryption on an unencrypted port

- New applications only need one port for both normal and SSL traffic

Primary user is Internet Print Protocol (IPP)

- Web browsers do not yet support this... but when they do:

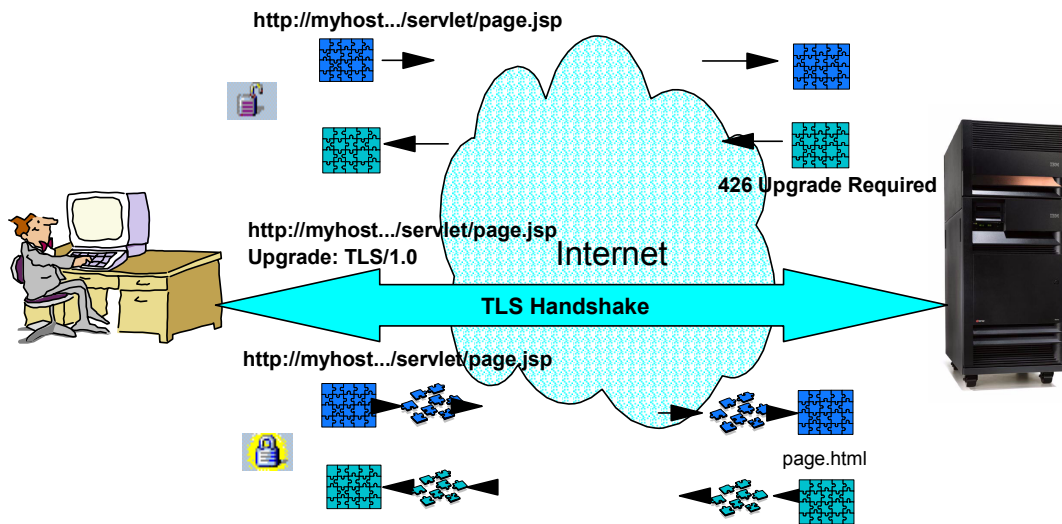


Figure 6-34 TLS upgrade

The historical practice of deploying HTTP over SSL3 has distinguished the combination from HTTP alone by a unique Uniform Resource Identifier (URI) scheme and the TCP port number. The scheme “http” meant the HTTP protocol alone on port 80, while “https” meant the HTTP protocol over SSL on port 443. Parallel well-known port numbers were similarly requested, and in some cases granted, to distinguish between secured and unsecured use of other application protocols (for example news and FTPS). This approach effectively divides in half the number of available well-known ports.

At the Washington D.C. Internet Engineering Task Force (IETF) meeting in December 1997, the Applications Area Directors and the Engineering Steering Group (IESG) reaffirmed that the practice of issuing parallel “secure” port numbers should be deprecated. The HTTP/1.1 Upgrade mechanism can apply Transport Layer Security [6] to an open HTTP connection.

In the nearly two years since, there has been broad acceptance of the concept behind this proposal, but little interest in implementing alternatives to port 443 for generic Web browsing. In fact, nothing in this memo affects the current interpretation of HTTPS: URIs. However, new application protocols built atop HTTP, such as the Internet Printing Protocol, call for just such a mechanism to move ahead in the IETF standards process.

The Upgrade mechanism also solves the “virtual hosting” problem. Rather than allocating multiple IP addresses to a single host, an HTTP/1.1 server uses the Host: header to disambiguate the intended Web service. As HTTP/1.1 usage has grown more prevalent, more Internet Service Providers (ISPs) are offering name-based virtual hosting, thus delaying IP address space exhaustion.

TLS (and SSL) have been degraded by the same limitation as earlier versions of HTTP. The initial handshake does not specify the intended host name, relying exclusively on the IP address. Using a cleartext HTTP/1.1 Upgrade: preamble to the TLS handshake, choosing the certificates based on the initial Host: header, allows ISPs to provide secure name-based virtual hosting as well.

6.4.3 Enabling SSL for the ADMIN instance

Enabling SSL support for the configuration GUI requires additional considerations. First add the following lines to the ADMIN customization include that is located in /QIBM/UserData/HTTPAdmin/conf/admin-cust.conf:

```
LoadModule ibm_ssl_module /QSYS.LIB/QHTTSPVR.LIB/QZSRVSSL.SRVPGM
Listen 2001
Listen 2010
SetEnv HTTPS_PORT 2010
<VirtualHost *:2010>
SSLEnable
SSLAppName QIBM_HTTP_SERVER_ADMIN
</VirtualHost>
```

Then enter the following CL command to register the ADMIN instance within the DCM environment:

```
CALL QHTTSPVR/QZHAPREG PARM('RegisterAppName' 'QIBM_HTTP_SERVER_ADMIN')
```

You can now access the DCM GUI and assign a server certificate to the QIBM_HTTP_SERVER_ADMIN application that you just registered.

For more information about setting up SSL for the HTTP Admin instance, refer to the iSeries Information Center at:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/ic2924/info/rzaie/rzaieconfigssladmin.htm>

6.4.4 SSL handshaking

Three versions of SSL, or protocols, are in use. SSL V2 and SSL V3 are standards published by Netscape. TLS V1 is an RFC (RFC2246) published and administered as are the other Internet standards. RFC2246 has been updated by RFC3546.

SSL V2 is old and not often used now. It does not support client authentication and has a number of known security weaknesses. SSL V3 is significantly different from SSL V2 and is now the protocol most commonly used. TLS V1 is a relatively new protocol, similar to SSL V3, that addresses some security and performance issues discovered in SSL V3.

Both the SSL clients and servers have a list of *ciphers*, known as the *cipher suite list* that they are willing to use. During the SSL handshake, the lists are compared and a cipher, normally the strongest, is chosen.

Tip: This section helps you understand how to configure these cipher lists in your HTTP Server (powered by Apache). The theory behind the example used in this section is based on the IBM Redbook *IBM @server iSeries Wired Network Security: OS/400 V5R1 DCM and Cryptographic Enhancements*, SG24-6168. This redbook is a *must read* for your work.

In a perfect world and with modern Web browsers, the negotiation of the suite of ciphers used is transparent to both the client and even the HTTP server administrator. But, in a business-to-consumer (B2C) environment, you cannot control the age and quality of the Web browser.

The handshake

When a client (that is, a Web browser) establishes an SSL session, the client sends a *Client Hello* message to the server. Among other things, this message contains all supported cipher suites that the browser can handle. A single cipher suite is a combination of an encryption protocol (that is, DES, RC4, AES), the encryption key length (40, 56, or 128), and a hash algorithm (SHA or MD5) that is used for integrity checking. When the server receives the Client Hello message, the server controls and decides which offered cipher suite it will use to secure this particular session.

With no SSLVersion or SSLCipherSpec directives specified, the server accepts anything. That is true for all newer browsers, because they support the handshake correctly. With some older browsers, there are some problems due to bugs in their implementation of the SSL protocol at that time.

To fix these bugs, you can force your clients to upgrade to a more modern version of Web browser. As mentioned earlier, however, in a B2C environment, this is not always an option. Or, you can control the order that the ciphers are selected during the SSL handshake to avoid know problems with older (buggy) Web clients.

To actually force a client to work with a specific encryption strength, you have the chance, via directives, to control what algorithms and key lengths your server will accept. If, for example, your client only supports 56-bit encryption and your server requires at least 128-bit, then the handshake fails. As a general rule, the SSLCipherSpec directives are good to limit the number of algorithms and key lengths to what you think is acceptable and secure. That, of course, depends on the content to be protected (how sensitive) and the client community you are dealing with.

An example

For the HTTP Server (powered by Apache), you have to add one SSLCipherSpec directive per cipher suite from top (preferred) to bottom.

The problem is that Microsoft's Internet Explorer's (IE) 56-bit SSL implementation has flaws in it. The flaws prevent it from negotiating with servers that use higher encryption than it does unless you disable a few ciphers on the server side. This bug was released with IE Version 5, and the only fix that Microsoft has for it is to install 128-bit encryption. To circumvent this problem in the HTTP Server (powered by Apache) configuration, use these directives:

```
SSLCipherSpec SSL_RSA_WITH_RC4_128_MD5
SSLCipherSpec SSL_RSA_WITH_RC4_128_SHA
SSLCipherSpec SSL_RSA_WITH_DES_CBC_SHA
SSLCipherSpec SSL_RSA_WITH_3DES_EDE_CBC_SHA
```

SSLCipherSpec SSL_RSA_EXPORT_WITH_RC4_40_MD5
SSLCipherSpec SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5

Attention: The third line SSLCipherSpec SSL_RSA_WITH_DES_CBC_SHA indicates use of DES with 56-bit encryption. The fourth line SSLCipherSpec SSL_RSA_WITH_3DES_EDE_CBC_SHA indicates use of 3DES with 168-bit encryption. Why would you want to choose a weaker cipher before choosing the stronger one? To solve a problem with an older browser.

You must be careful to not “oversell” to your clients the strength of your encryption on your site if you make these kinds of server-side configuration changes.

Through the IBM Web Administration for iSeries tool, you can configure directives to control:

- ▶ The cipher suites that the server accepts for SSL and TLS sessions
- ▶ The secure protocol version that the server accepts
- ▶ SSL/TLS timeout values
- ▶ SSL caching

To configure the directives for the context that you want to protect, click **Security** and then select the **SSL Advanced** tab (Figure 6-35). To activate the SSL Advanced tab, you need to enable SSL on the SSL with Certificate Authentication tab.

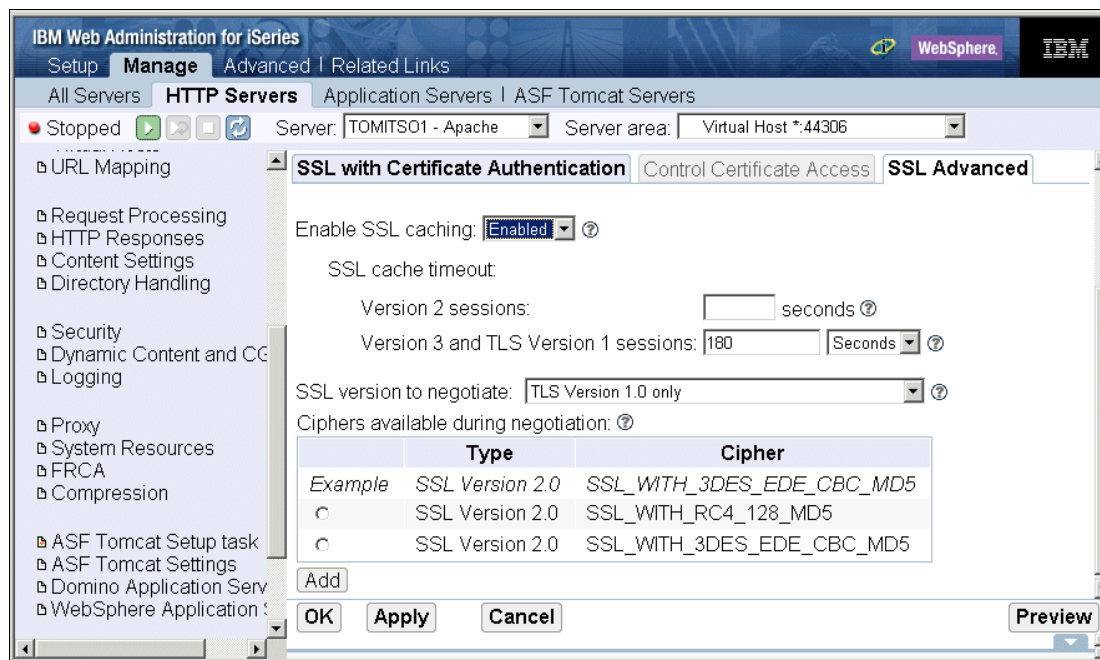


Figure 6-35 SSL Advanced tab

6.4.5 Client-side digital certificates

Client-side digital certificates are an advanced means of user authentication. A user certificate issued by the server is installed in the browser and used to verify the end user's identity.

Implementation

This section explains how to protect one of the IFS folders using client authentication:

1. From the Server area list, select the SSL-secured virtual host as the active context. In this example, we select **Virtual Host *:44306**.
2. In the left pane, select **Security**.
3. Select the **SSL with Certificate Authentication** tab.
4. In the SSL with Certificate Authentication page (Figure 6-36), under Client certificates when establishing the connection, select **Require client certificate for connection**. Click **OK**.

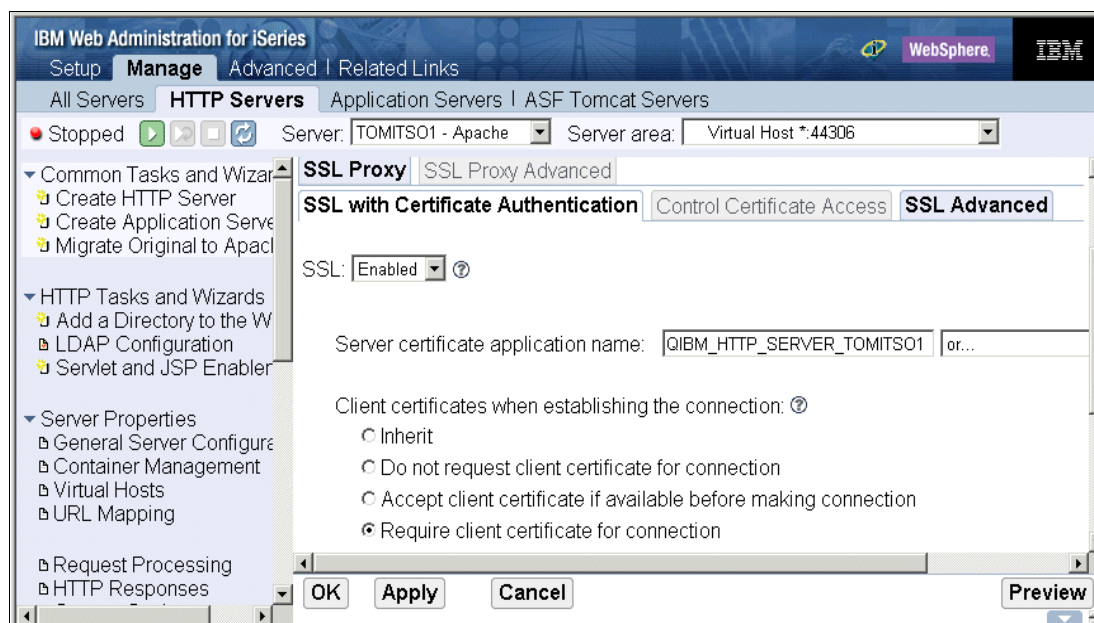


Figure 6-36 SSL Client Authentication for Virtual Host

5. The folder that we want to protect is served by a directory directive inside our virtual host context.
 - a. Ensure that you still have the **Virtual Host *:44306** context selected in the Server area list.
 - b. Start the Add a Directory to the Web wizard.
 - c. Follow the wizard pages to serve a new directory.

6. Looking at the Server area list, you can see the new directory listed below the virtual host, as shown in Figure 6-37. From the Server area list, select the new directory as the active context.

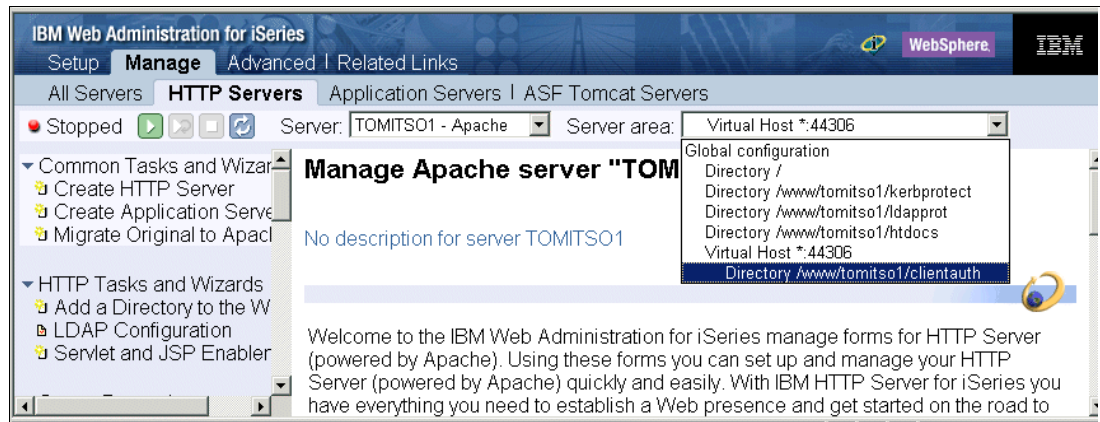


Figure 6-37 Directory selection from Server area list

7. In the left pane, click **Security**.
8. Select the **Control Certificate Access** tab.
9. On the Control Certificate Access page (Figure 6-38), complete these tasks:
 - a. Under Verification of expiration and trusted root, select **Verify client certificate**.
 - b. Under Client certificate authentication, select **Use client certificate**.
 - c. Scroll down. Select **Client certificate must be associated with OS/400 user profile**.

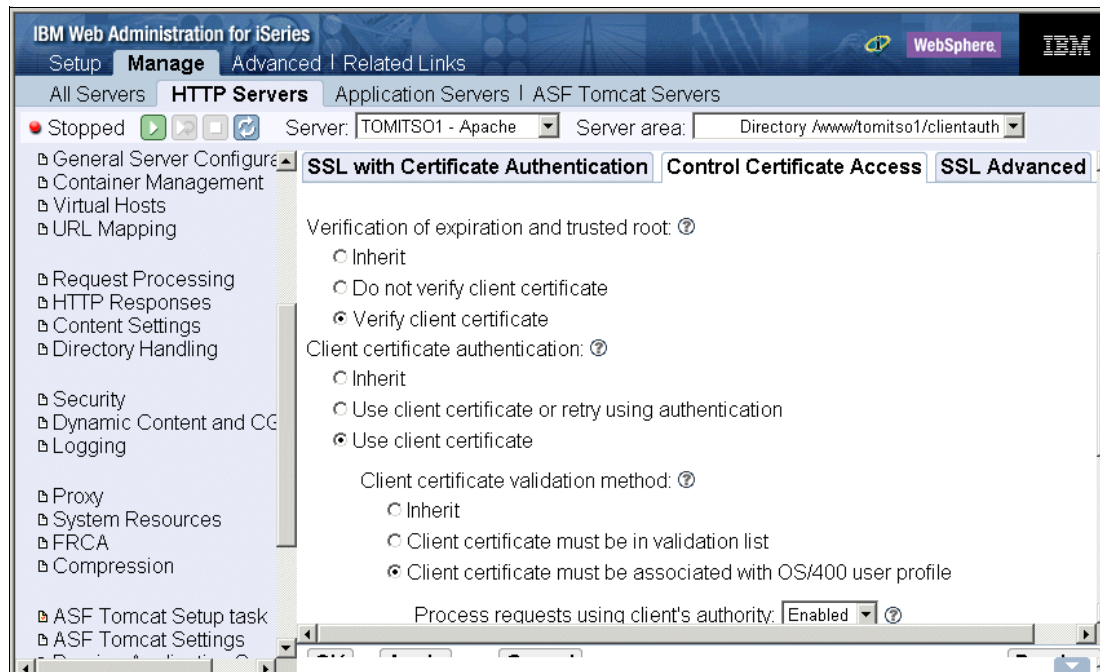


Figure 6-38 SSL client authentication

- d. From the Process requests using client's authority list, select **Enabled**.

This option enforces OS/400 object level access authority settings. If you set this value to *Disabled*, access to the protected resources is, by default, performed under the QTMHHTTP profile for static pages and QTMHHTTP1 profile for CGI programs unless the ServerUserID directive specifically names another user profile.

- e. Click **Apply**.

10. In the right panel, select the **Control Access** tab.

11. On the Control Access page, complete these steps:

- a. Select **All authenticated users (valid user name and password)**.
- b. Scroll down to the Control access policy section. Select **Control access based on where or from whom the request originates**.

Important: Selecting the Control access based on where or from whom the request originates option adds the Satisfy Any directive to the HTTP configuration. The Any value for the satisfy directive is required for SSL client authentication to work. If you select Inherit (which defaults to ALL) or select Control access based on where and from whom the request originates, the All value for the Satisfy directive requires a valid client certificate *and* authentication via user and password.

- c. Click **OK** to save your settings.

12. Restart your server.

This configuration requires a valid client certificate for access to the new folder. It also forces the server to access protected data as the user for whom that certificate was issued. This powerful capability of the HTTP Server (powered by Apache), also known as *profile swapping*, is further proof of the granularity and versatility of the HTTP Server (powered by Apache) implemented on the iSeries server.

Alternatively, you can configure SSL client authentication to authenticate client certificates that are stored in validation lists or client certificates that meet certain criteria in the DN. For example, you can allow users access to a protected resource where the certificate subject DN contains an organization (attribute of IBM and the issuer DN's common name attribute contains VeriSign).

6.5 Proxy server: Protecting direct access

Proxy servers are deployed on a network for two key purposes: security and performance. A proxy can be used to monitor and filter inbound and outbound requests. Or it can be used as a single point of access for communications with untrusted networks. Proxies can also dramatically improve HTTP response times by serving documents from a local cache (see 10.3.2, "HTTP Server (powered by Apache) proxy cache" on page 239). This effectively reduces network traffic, bandwidth occupation, and Central Processing Unit (CPU) load (depending on the type of request being served).

This section focuses on the two mainstream proxy implementations: the *forward proxy* and the *reverse proxy*. Both can be implemented as virtual hosts or stand-alone servers. Apache proxy can also be configured as part of a proxy chain by specifying to which server the requests will be relayed.

Table 6-3 offers a brief overview of the configuration steps with the resulting Apache directives.

Table 6-3 Proxy configuration overview

Goal	GUI configuration steps	Apache final configuration file
Enable proxy support. See 6.5.1, “Forward proxy” on page 143, for an example implementation.	Under the Forward Proxy tab, select Enable . Optionally define how Via headers will be handled.	2 LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM 3 LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM 4 LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM 5 LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM 24 ProxyRequests On 25 ProxyVia off
Proxy mapping rules (reverse proxy or proxy chain only). See 6.5.2, “Reverse proxy”, for an example implementation.	Under the Reverse Proxy or the Proxy Chaining tab, define mapping rules (Pass file access requests to remote servers) or filters (Block URL requests).	25 AllowCONNECT 443 26 AllowCONNECT 563 27 ProxyNoConnect Off 28 ProxyReceiveBufferSize 0 29 ProxyPass /webprojects/ http://www.myco.com/projects/ 30 ProxyPassReverse /webprojects/ http://www.myco.com/projects/

6.5.1 Forward proxy

A forward proxy fetches content from another server, allowing clients to reach a network to which they wouldn’t otherwise have access. Figure 6-39 demonstrates the role of a forward proxy in an environment where clients on a private intranet do not have direct access to the Internet.

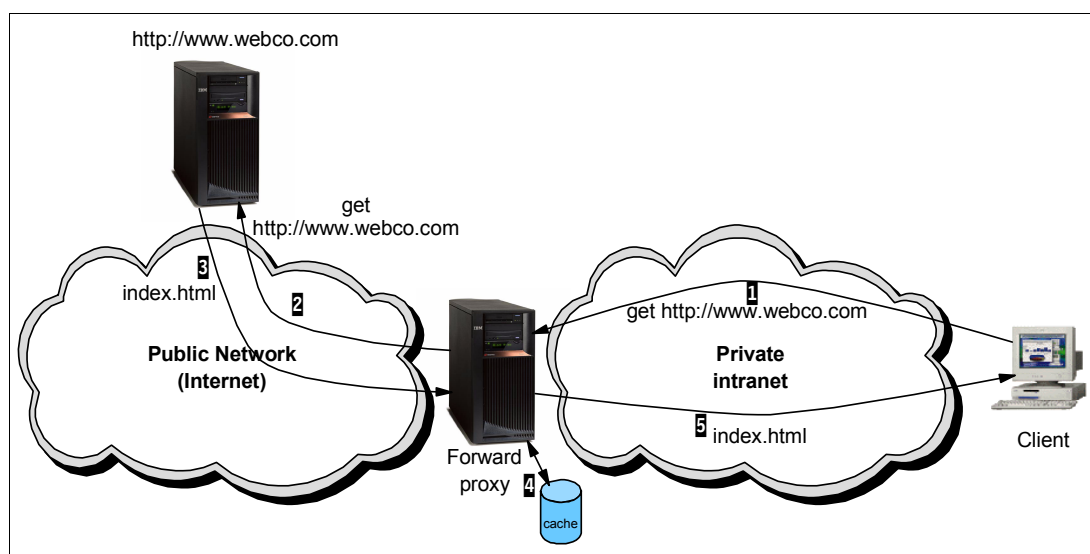


Figure 6-39 Forward proxy: Example network

In this configuration, the clients send all outbound HTTP requests to the forward proxy, as indicated by 1 in Figure 6-39. The proxy checks the request against security restrictions and then looks for a valid copy of the requested document in the local cache. If the document can be retrieved from the cache, the proxy poses as the destination server itself, and serves it to

the client. Otherwise the proxy establishes a connection to the www.webco.com server, indicated by 2, and retrieves an updated copy of the document, indicated by 3. The document is (optionally) stored in the local cache (see 4) and sent to the requestor (noted by 5). Note that from the client's point of view, the proxy *is* the Web server itself, and no other system appears to be involved in the transaction.

Implementation

Figure 6-40 and the following steps guide you through the configuration for proxy support activation:

1. From the Server area list, select **Global Configuration**.
2. In the left pane, under Server Properties, select **Proxy**.
3. Select the **Forward Proxy** tab.
4. On the Forward Proxy page, complete these tasks:
 - a. Under Forward proxy capabilities, select **Enabled**.
 - b. If necessary, specify a default domain for unqualified requests. This suffix is used whenever a fully qualified name is not used in the client's request. In the example in Figure 6-40, the ibm.com domain is appended to unqualified requests.
 - c. Add the ports that you want to allow for SSL traffic through the proxy server. For example, if a user accesses a Web page via HTTP and clicks a link that switches to HTTPS, the SSL connection is not established when the SSL port is not listed in the CONNECT method for HTTPS requests list. The well-known port for HTTPS is port 443. As shown in Figure 6-40, we added port 443.

Note: Remember to add every port that should be allowed for HTTPS requests through the proxy server to the list of ports for the CONNECT method. HTTPS requests for ports that are not defined for the CONNECT method will fail.

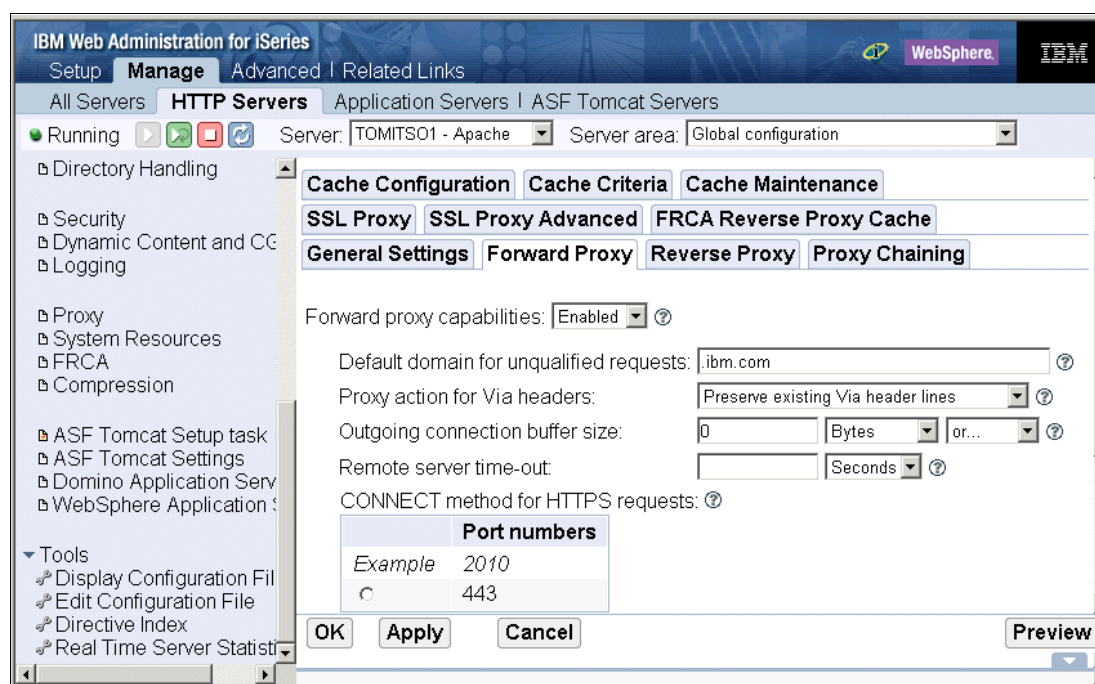


Figure 6-40 Forward proxy settings

- d. Scroll down on the Forward Proxy page. You see a section named Incoming URL requests to block. This section allows you to define words, host names, and domain names.

Requests to sites whose URLs contain matched words, hosts, or domains are blocked by the server. At startup, the server attempts to determine list item IP addresses, that may be host names, and records them for a match test. The values entered can be any part of the host address part of the URL. The values are not checked against any other part in the URL, such as a path or file name. For example, if you want to block all requests to all sites at the myco.com domain, such as www.products.myco.com, you can add the entry myco.com. If the MYCO corporation operates sites under the .com, .org, and .net domains, you can enter just the value myco, as shown in Figure 6-41, to block requests to any page of the MYCO corporation.

If required, click the **Add** button to enter values for sites to be blocked.

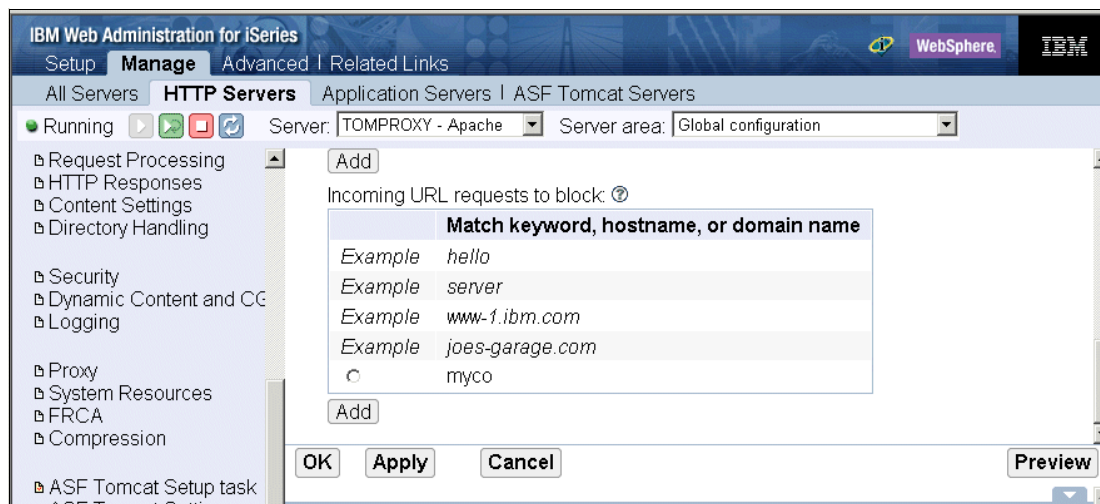


Figure 6-41 Forward proxy settings: Incoming URL requests to block

- e. Click **Apply**.
 - f. Click **OK**.
5. Restart your server and test it.

6.5.2 Reverse proxy

Reverse proxy is the same as a forward proxy, except that requests from outside of the firewall to the proxy are allowed.

Tip: Another reverse proxy associated with your HTTP Server (powered by Apache) is Fast Response Cache Accelerator (FRCA). See 10.6, “Fast Response Cache Accelerator” on page 281, for more details and a configuration example of using FRCA as a reverse proxy.

A reverse proxy is another common form of a proxy server. It is generally used to pass requests from the Internet, through a firewall, to isolated, private networks. It is used to prevent Internet clients from having direct, unmonitored access to sensitive data residing on content servers on an isolated network or intranet. If caching is enabled, a reverse proxy can also reduce network traffic by serving cached information rather than passing all requests to actual content servers. Reverse proxy servers may also balance workload by spreading requests across a number of content servers.

An advantage of using a reverse proxy is that Internet clients do not know their requests are being sent to and handled by a reverse proxy server. This allows a reverse proxy to redirect or reject requests without making Internet clients aware of the actual content server (or servers) on a protected network.

A reverse proxy server first checks to make sure a request is valid. If a request is not valid, or not allowed (blocked by the proxy), it does not continue to process the request resulting in the client receiving an error or a redirect. If a request is valid, a reverse proxy may check if the requested information is cached. If it is, the reverse proxy serves the cached information. If it is not, the reverse proxy requests the information from the content server and serves it to the requesting client. It can also cache the information for future requests.

A quick and easy demonstration of the power of a reverse proxy is shown in Figure 6-42.

Tip: Compare the network diagrams in Figure 6-39 on page 143 and Figure 6-42. One of the major points that you should notice is that we switched the label on the two different networks. That is, for the forward proxy, the client is connected to an internal intranet and needs access to the Internet. For the reverse proxy, the client is connected to a public network and needs access to a content server found someplace in the internal intranet. One of the features of reverse proxy is that you can hide, from the remote clients, the internal IP addresses and naming of your intranet and applications.

In this configuration, the clients send all HTTP requests to the iSeries reverse proxy server, as indicated by **1** in Figure 6-42, at the public host and domain name of `www.webco.com`. The reverse proxy checks the request against security restrictions and then looks for a valid copy of the requested document in the local cache. If the document can be retrieved from the cache, the reverse proxy server serves it to the client. Otherwise the reverse proxy establishes a connection to the as23 content server, as indicated by **2**, and retrieves an updated copy of the document, indicated by **3**. The document is (optionally) stored in the local cache (see **4**) and sent to the requestor, as shown by **5**. Note that from the client's point of view, the reverse proxy *is* the Web server itself, and no other system appears to be involved in the transaction.

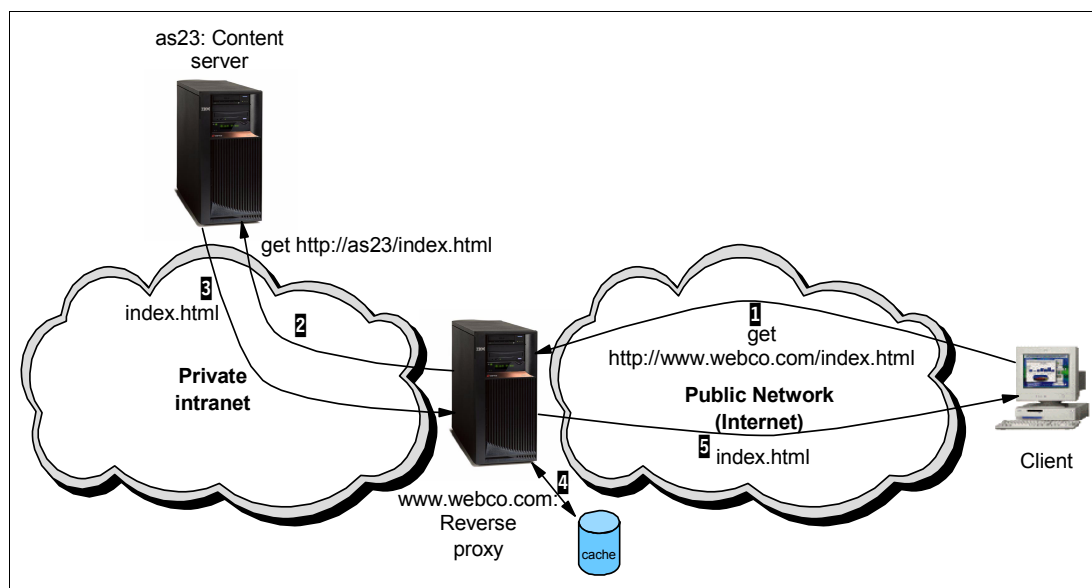


Figure 6-42 Reverse proxy: Example network

Note: The reverse proxy server as described in this section supports HTTP connections only. If you want to SSL-protect the entire traffic from the client browser through the proxy to the content server, configure the SSL proxy as explained in 6.5.3, “SSL proxy” on page 149.

Implementation

To configure a simple reverse proxy scenario, we use two iSeries servers.

1. On as23 (your content server), create an HTTP Server (powered by Apache) with the parameters listed in Table 6-4. For this simple scenario, do *not* add any additional configuration directives to the content server to support reverse proxy.

Table 6-4 Reverse proxy: Configuration basics for as23 content server

Create HTTP Server wizard parameter	Value
Server name	PBABASIC02
Server root	/tcp52d02/basicConfig
Document root	/tcp52d02/basicConfig/ITSOco
On which IP address and TCP/IP port do you want your server to listen?	IP address: 10.5.92.30 Port: 8002
Do you want your new server to use an access log?	Yes

2. On www.webco.com (your reverse proxy server), create an HTTP Server (powered by Apache) with the parameters in Table 6-5.

Table 6-5 Reverse proxy: Configuration basics for www.webco.com reverse proxy server

Create HTTP Server wizard parameter	Value
Server name	PBARPRXY02
Server root	/tcp52d02/rprxy
Document root	/tcp52d02/rprxy/itsoco
On which IP address and TCP/IP port do you want your server to listen?	IP address: all Port: 8002
Do you want your new server to use an access log?	Yes

3. Add the necessary proxy directives to the HTTP Server (powered by Apache) PBARPRXY02 server configuration so that for all incoming URI path requests traffic are sent to the content server on as23. Select the **Manage** tab.
4. From the Server list, select **PBARPRXY02**.
5. From the Server area list, select **Global configuration**.
6. In the left pane, select **Proxy**.
7. In the right panel, select the **Reverse Proxy** tab.
8. From the Reverse proxy capabilities list, select **Enabled**. This expands your options for this page as shown in Figure 6-43. In addition, enabling reverse proxy causes the GUI to add the following LoadModule directives to the global context to support proxy:

```
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

- Under Proxy requests to remote servers, click **Add** to add a new row into the table. Add one row for the redirected requests and another for client requests. The order does not matter. This adds the following directives to your httpd.conf configuration file:

```
ProxyPassReverse / http://10.5.92.30:8002/
ProxyPass / http://10.5.92.30:8002/
```

- After you enter the information for each entry, click **Continue** to save the entry into the list.

Now all of the get requests for this instance of the HTTP Server (powered by Apache) running on www.webco.com are redirected to the fully qualified URL of the content server at http://10.5.92.30 on port 8002. You can also specify individual request paths to be answered by a different content server behind the reverse proxy. When defining the Redirected requests option, headers in response documents are adjusted in the event that a "Redirect" is issued by the remote server. This allows clients to remain unaware of any transformation of the requests even if remote servers redirect the proxy.

Note: We used the IP address of the content server as23 for performance reasons. The host name (as23) can be used instead. If you use a name, such as as23, your reverse proxy server www.webco.com must have the ability to resolve the name to an IP address using either a local host table or a split Domain Name System (DNS) server. For an example of how to configure a split DNS on the iSeries, see *iSeries IP Networks: Dynamic!*, SG24-6718.

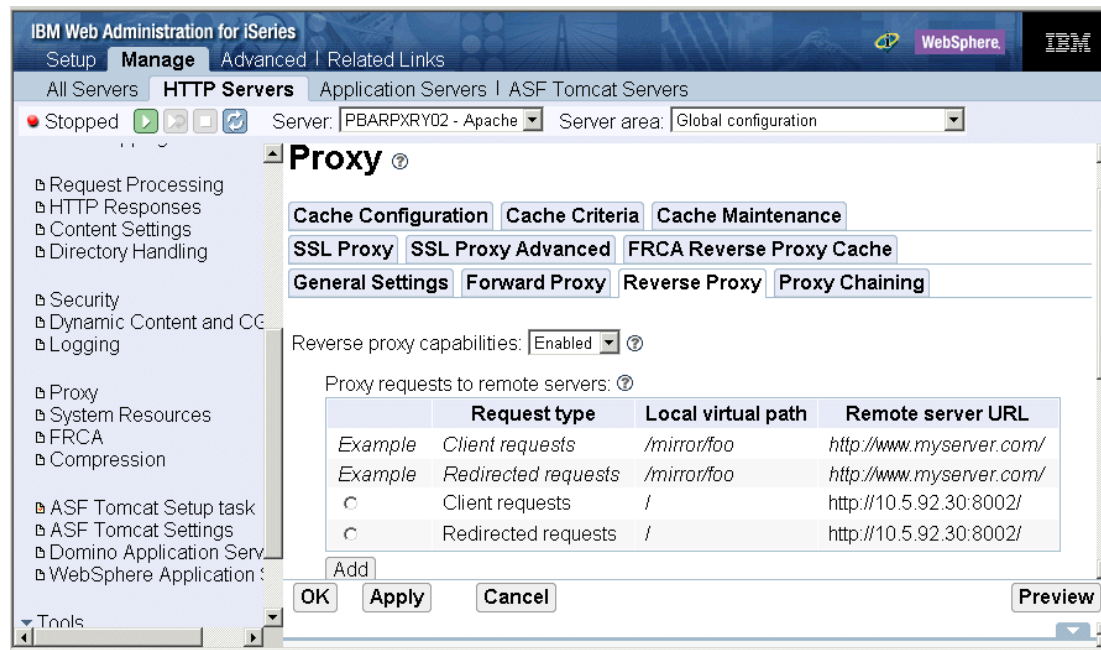


Figure 6-43 Reverse proxy: GUI configuration

- Click **OK**.
- Start both servers. The following client get request is forwarded to the content server:

http://www.webco.com/index.html

It is sent using the following URL:

http://10.5.92.30:8002/index.html

This processing is transparent to the client.

6.5.3 SSL proxy

The SSL proxy is typically used as a reverse proxy that supports SSL for the connection from the client browser through the proxy to the content server. In fact, two connections are established. The first HTTPS connection is established from the browser to the proxy server. This connection terminates at the proxy, which in turn establishes a second HTTPS connection from the proxy to the content server. You can also configure the proxy server to always establish a SSL connection from the proxy to the content server, but to allow non-protected (HTTP) and protected (HTTPS) connections from the client to the reverse proxy.

Note: As stated earlier, the SSL proxy is used for establishing a protected session from the client through the proxy server to the content server in a reverse proxy environment. For forward proxy protected sessions, the CONNECT request method is used as described in 6.5.1, “Forward proxy” on page 143.

In this configuration, the clients send all HTTP requests to the iSeries reverse proxy server, as indicated by **1** in Figure 6-44, at the public host and domain name of www.webco.com. This time, the client accesses sensitive information and, therefore, needs a protected session from the client to the content server. The first SSL connection is established between the client and the proxy at www.webco.com. The reverse proxy checks the request against security restrictions and then looks for a valid copy of the requested document in the local cache. If the document can be retrieved from the cache, the reverse proxy server serves it to the client. Otherwise the reverse proxy establishes a second SSL connection to the fra822 content server, as indicated by **2**, and retrieves an updated copy of the document, indicated by **3**. The document is (optionally) stored in the local cache (see **4**) and sent to the requestor, as shown by **5**. Note that from the client's point of view, the reverse proxy *is* the Web server itself, and no other system appears to be involved in the transaction.

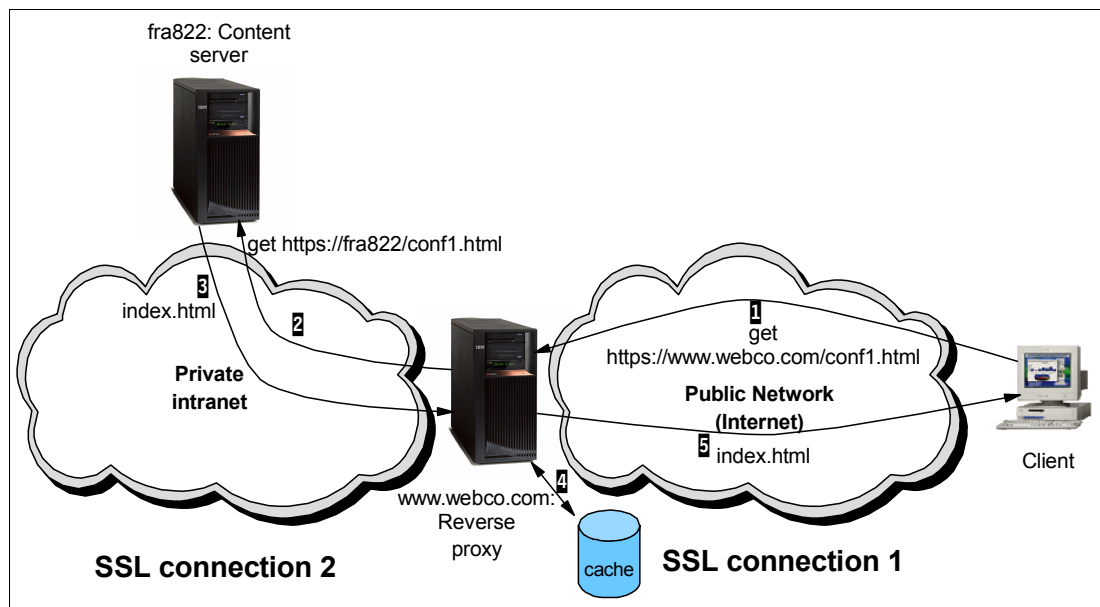


Figure 6-44 SSL proxy environment

The setup of the SSL proxy involves some configuration tasks to be performed via the DCM. If you are not familiar with DCM, you may want to consult the IBM Redbook *IBM @server iSeries Wired Network Security*, SG24-6168.

Implementation

The following steps illustrate the implementation steps for the SSL reverse proxy support:

1. On fra822 (your content server), create an HTTP Server (powered by Apache) with the parameters in Table 6-6. For this simple scenario, do *not* add any additional configuration directives to the content server to support reverse proxy.

Table 6-6 Reverse SSL proxy: Configuration basics for fra822 content server

Create HTTP Server wizard parameter	Value
Server name	TOMSERV1
Server root	/www/tomserv1/
Document root	/www/tomserv1/htdocs
On which IP address and TCP/IP port do you want your server to listen?	IP address: All Port: 80
Do you want your new server to use an access log?	Yes

2. Using the configuration steps from 6.4.1, “Enabling SSL” on page 127, configure your content server TOMSERV1 to allow SSL connections on port 443 and use DCM to assign a server certificate to the content server.
3. On www.webco.com (your reverse proxy server), create an HTTP Server (powered by Apache) with the parameters in Table 6-7.

Table 6-7 Reverse SSL proxy: Configuration basics for www.webco.com reverse proxy server

Create HTTP Server wizard parameter	Value
Server name	TOMSSLPROX
Server root	/www/tomsslprox
Document root	/www/tomsslprox/htdocs
On which IP address and TCP/IP port do you want your server to listen?	IP address: all Port: 443
Do you want your new server to use an access log?	Yes

Note: The proxy server is configured to allow only SSL connections from the client to the server. Therefore, port 443 was selected at the Create HTTP Server wizard.

4. Select the **Manage** tab.
5. From the Server list, select **TOMSSLPROX**.
6. Create a virtual host and enable SSL for the virtual host as explained in 6.4.1, “Enabling SSL” on page 127. Use the secure application name QIBM_HTTP_SERVER_TOMSSLPROX. Use DCM to assign a server certificate to the HTTP application name. This certificate represents the content server on the public network. The common name of the certificate should be www.webco.com.
7. Add the necessary proxy directives to the HTTP Server (powered by Apache) TOMSSLPROX server configuration so that for all incoming URI path requests traffic are sent to the content server on fra822. From the Server area list, select **Global configuration**.
8. In the left pane, select **Proxy**.
9. In the Proxy panel, click the **Reverse Proxy** tab.

10. On the Reverse Proxy page, complete these steps:

- a. From the Reverse proxy capabilities list, select **Enabled**. For more information about the parameter of the reverse proxy function, see 6.5.2, “Reverse proxy” on page 145.
- b. Under Proxy requests to remote servers, click **Add** to add a new row into the table. Add one row for the redirected requests and another for client requests. The order is not important. This adds the following directives to your httpd.conf configuration file:

```
ProxyPassReverse / https://10.164.96.84/  
ProxyPass / https://10.164.96.84/
```

Note that the protocol is HTTPS and not HTTP. Using these directives, the server always establishes a protected session from the proxy to the content server.

- c. After entering the information for each entry, click **Continue** to save the entry into the list.

11. Click the **SSL Proxy** tab.

12. On the SSL Proxy page (Figure 6-45), complete these tasks:

- a. From the SSL Proxy list, select **Enabled**.
- b. For Proxy server certificate application name, select **QIBM_PROXY_SERVER_TOMSSLPROX**. Note that the application name is different from the name used in the virtual host section. It uses the word PROXY instead of HTTP. This is important to know when assigning certificates in the DCM.
- c. For Content server certificate required by proxy server, select **Require non-expired and trusted root certificate**. This options ensures that the content server must present a certificate during the SSL handshake that is not expired and is marked as trusted in the DCM.

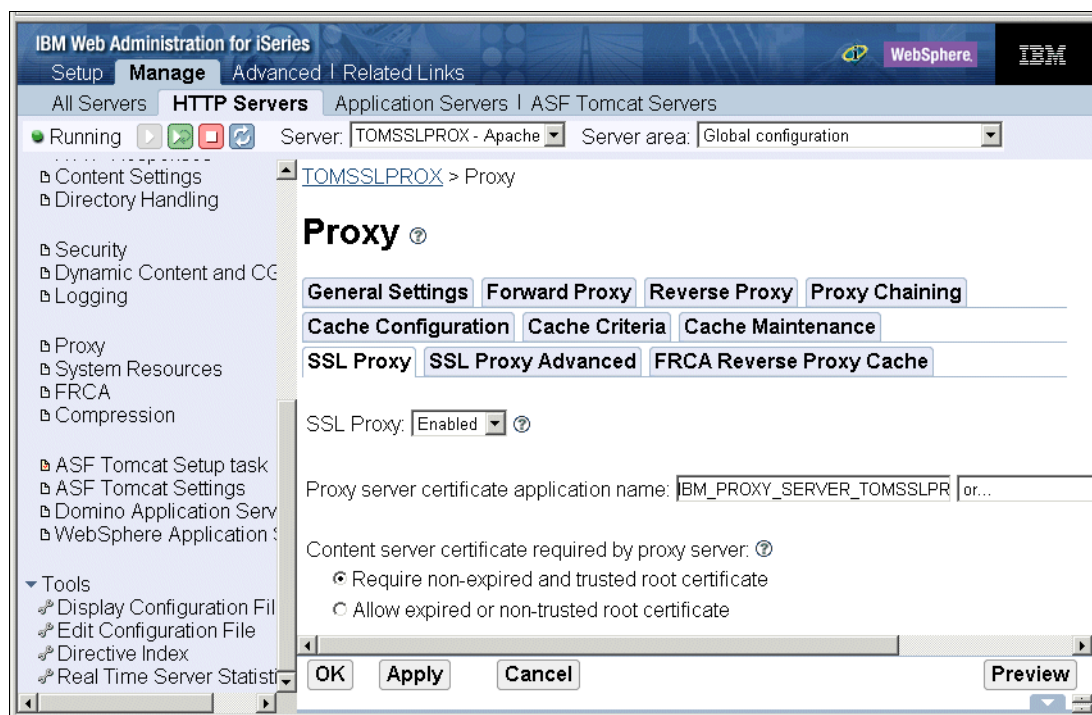


Figure 6-45 SSL Proxy tab

You could further refine your SSL settings on the SSL Proxy Advanced tab. The settings on this tab allow you to select the SSL protocols and cipher suites that are supported for the connection from the proxy server to the content server.

13. Click **Apply** and then **OK**. Note that caching has not been enabled yet.

In the remaining steps of the setup, you use the DCM to define the CA trust for and assign a certificate to the proxy server.

Important: As mentioned earlier, the SSL proxy configuration Require non-expired and trusted root certificate option requires that the proxy application trusts the CA that issued the content server's certificate. This means that the list of enabled CA certificates in the *SYSTEM store on the proxy server must contain the CA certificate of the content server's certificate. Since the content server is typically installed in an intranet environment, it is likely that the server certificate was issued by a local or private CA rather than a well-known CA, such as VeriSign. If the local CA is operated on the content server, import the CA certificate on the proxy server:

1. Send the CA.CACRT file by FTP from the content server directory /QIBM/UserData/ICSS/Cert/Download/CertAuth to an IFS directory on the proxy server.
2. Using DCM, open the *SYSTEM certificate store on the proxy server.
3. Click **Fast Path** and then **Work with CA certificates**.
4. Click **Import** at the bottom of the CA certificates list, specify the path and file name of the file sent by FTP, and enter a label name for the certificate to be imported.

1. From the IBM Web Administration for iSeries GUI, click the **Related Links** tab and launch DCM.
2. Click **Select a Certificate Store** and open the *SYSTEM certificate store.
3. Click **Fast Path** and then **Work with server applications**.
4. Select the application **QIBM_PROXY_SERVER_TOMSSLPROX** and click **Work with Application**.
5. Click **Define CA Trust List** and select the CA certificate of the CA that issued the content server certificate (the certificate that is assigned on the content server to the content server's HTTP server instance).
6. In the Define CA Trust List window, click **OK** to save the new trust list.

- Click **Cancel** to return to the Work with Application window (Figure 6-46).

Note: You do not need to assign a certificate to the proxy application if the content server is configured for server authentication only. However, if the content server requires client authentication, you must assign a certificate to the proxy server application, since the proxy server acts as a client when connecting to the content server. Otherwise, the SSL handshake between the content server and proxy server fails.

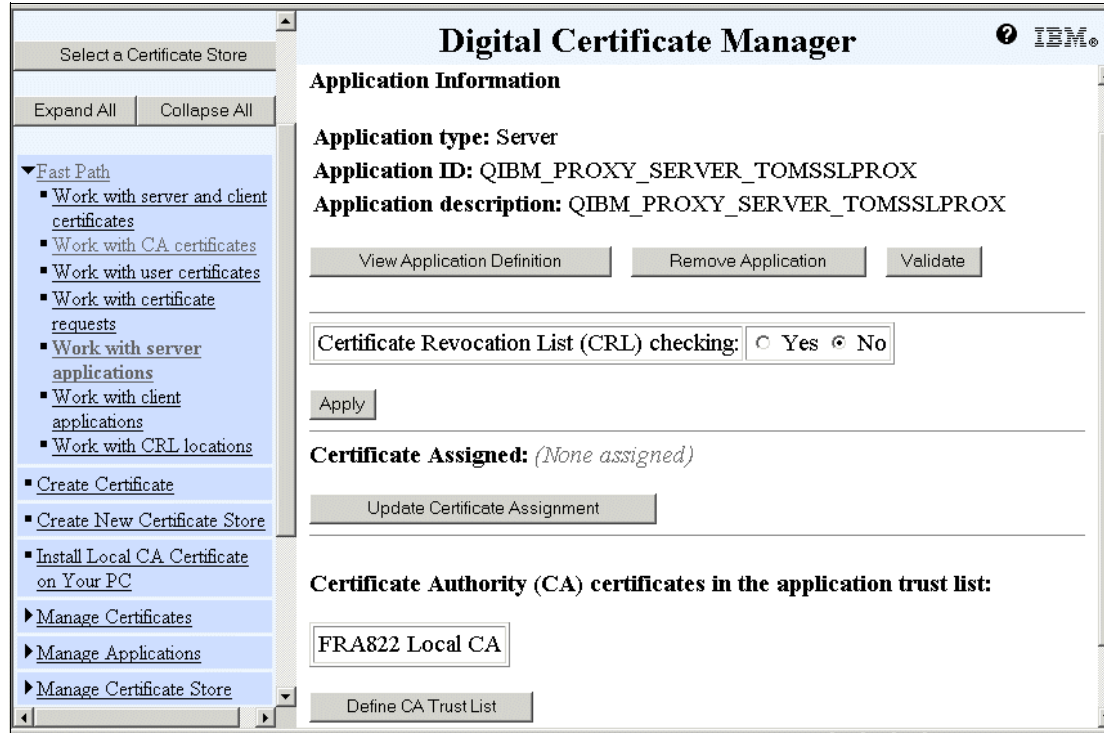


Figure 6-46 DCM Work with Application

- Start your proxy server and test your connection using the following URL:

`https://www.webco.com`

Since the proxy server instance is configured to accept port 443 only and this port is enabled for SSL in the virtual host context, only HTTPS connections can be established from the client to the reverse proxy server. If you want to enable both HTTP and HTTPS connections from the client to the proxy server, add an additional port, such as 80, to the global configuration context.

The HTTP configuration directives of the SSL proxy in this scenario are shown here, with SSL proxy-related directives in bold:

Configuration originally created by Create HTTP Server wizard on Mon Sep 20 14:48:42 CEST 2004

```
LoadModule ibm_ssl_module /QSYS.LIB/QHTTSPVR.LIB/QZSRVSSL.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_connect_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
Listen *:443
DocumentRoot /www/tomsslprox/htdocs
```

```
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes
-MultiViews
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log combined
LogMaint logs/access_log 7 0
LogMaint logs/error_log 7 0
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\." force-response-1.0
SetEnvIf "User-Agent" "Java/1\." force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\." force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\." nokeepalive
SetEnvIf "User-Agent" "MSIE 4\." force-response-1.0
SSLProxyEngine On
SSLProxyAppName QIBM_PROXY_SERVER_TOMSSLPROX
ProxyPass / https://10.164.96.84/
ProxyPassReverse / https://10.164.96.84/
<Directory />
    Order Deny,Allow
    Deny From all
</Directory>
<Directory /www/tomsslprox/htdocs>
    Order Allow,Deny
    Allow From all
</Directory>
<VirtualHost *:443>
    SSLEngine On
    SSLAppName QIBM_HTTP_SERVER_TOMSSLPROX
</VirtualHost>
```

6.5.4 Proxy chaining

A proxy chain uses two or more proxy servers to assist in server and protocol performance and network security. Proxy chaining is not a type of proxy, but a use of reverse and forward proxy servers across multiple networks.

In addition to the benefits to security and performance, proxy chaining allows requests from different protocols to be fulfilled in cases where, without chaining, such requests are not possible or permitted. For example, a request using HTTP is sent to a server that can only handle FTP requests. For the request to be processed, it must pass through a server that can handle both protocols, which is accomplished by using proxy chaining. It allows the request to pass from a server that cannot fulfill such a request (perhaps due to security or networking issues or its own limited capabilities) to a server that can.

The first proxy server in a chain checks to make sure a request is valid. If a request is not valid, or not allowed (blocked by the proxy), it rejects the request. This results in the client receiving an error or being redirected. If a request is valid, the proxy may check if the requested information is cached and simply serve it from there. If the requested information is not in cache, the proxy passes on the request to the next proxy server in the chain.

This server can also fulfill, forward, redirect, or reject the request. If it forwards the request, then it too passes on the request to another proxy server. This process is repeated until the request reaches the last proxy server in the chain. The last server in the chain is required to handle the request by contacting the content server, using the required protocol, to obtain the information. The information is relayed back through the chain until it reaches the requesting client. Each proxy server in the chain may cache the information for future requests.

Reasons for passing requests through a proxy chain vary. For example, you may use proxy chaining to pass information through multiple networks where a client on one network cannot communicate directly with a proxy server on a different network, and it needs a second proxy to relay its requests. You may also use it to cache information in multiple locations or to allow certain protocols to be used outside a firewall which cannot be allowed through a firewall.

6.6 For more information

Refer to the following resources for additional information:

- ▶ *IBM @server iSeries Wired Network Security: OS/400 V5R1 DCM and Cryptographic Enhancements*, SG24-6168
- ▶ *AS/400 Internet Security: Developing a Digital Certificate Infrastructure*, SG24-5659
- ▶ *Implementation and Practical Use of LDAP on the IBM @server iSeries Server*, SG24-6193
- ▶ CERT/CC at Carnegie Mellon University offers an excellent source for security news, alerts, and papers:
<http://www.cert.org>
- ▶ HTTP Server (powered by Apache) uses module `mod_ibm_ssl` for all the authentication and encryption for SSL and TLS. You can find the manual for this module on the Web at:
http://publib.boulder.ibm.com/infocenter/iseriess/v5r3/ic2924/index.htm?info/rzaie/rzaiemod_ibm_ssl.htm
- ▶ For more information about proxy support by the HTTP Server (powered by Apache) see the Documentation Center. Go to the following Web site and select **Documentation**:
<http://www.ibm.com/servers/eserver/iseriess/software/http/docs/doc.htm>



Serving dynamic data

Maintaining static Hypertext Markup Language (HTML) pages can be easy and quite inexpensive, but static pages cannot cover all of your Web serving needs. Any time the published content needs to be tailored on the end user's input, the Web page has to be generated on the fly. Serving dynamic data from your HTTP server can be accomplished in several different ways, depending on your needs, your programming skills, and the complexity of the task at hand. The HTTP Server (powered by Apache) supports the most popular techniques generally available for this purpose such as:

- ▶ Server-side includes (SSI)
- ▶ Everything dynamic with Common Gateway Interface (CGI) support
- ▶ Net.Data: A ready-made scripting tool

Note: Notably missing from the above list are Hypertext Preprocessor (PHP) and Perl.

PHP is a server-side, cross-platform, HTML-embedded scripting language. Maybe the closest comparison to PHP is Net.Data as defined by IBM. Net.Data was created by IBM to “solve” the need for a server-side scripting language for its HTTP servers. PHP was created in the open-source community at about the same time to “solve” the need for a server-side scripting language for the Apache server.

IBM has not brought PHP to your HTTP Server (powered by Apache) for iSeries. That is why we have included Appendix A, “Bringing PHP to your iSeries server” on page 387, to demonstrate how you can bring PHP to your iSeries yourself as a CGI running in Portable Application Solution Environment (OS/400 PASE). The Rochester Development Lab is aware of this PHP requirement.

Perl too was made available for the iSeries by ingenious individuals. Yet it is also not directly supported by IBM. For more information about Perl for your iSeries, go to:

- ▶ <http://www.iseries.ibm.com/tstudio/workshop/tiptools/perl.htm>
- ▶ <http://www.cpan.org/ports/index.html#os400/>

Each of these techniques can be equally fast and powerful if it is properly employed. This chapter provides an overview of the different techniques and some examples.

As your Web publishing needs become more and more advanced, your programs will also grow in complexity. This may cause you some serviceability headaches. Most of the limitations you encounter can be overcome with an additional coding effort. Eventually you reach the point where a more powerful and efficient solution is needed.

Advanced implementations, such as transactional applications and solutions for On Demand Business, require more powerful, high-level tools such as Tomcat, WebSphere Application Server, WebSphere Commerce Suite, or equivalent third-party products. Servlets, JavaServer Pages (JSP), Enterprise JavaBeans (EJB), and other leading-edge Internet technologies are supported by these products. See Chapter 9, “Web application serving” on page 191, for more information about Apache support of these advanced technologies.

7.1 Server-side includes

Server-side includes are the simplest way to add dynamic content to a Web site. A set of directives is embedded in the HTML code and is interpreted by the server before the document is sent to a client. SSI can be used to trigger a CGI program or return information about documents or the value of environment variables, as shown in Figure 7-1. In a simple sense, SSI allows for character substitution from within an HTML document.

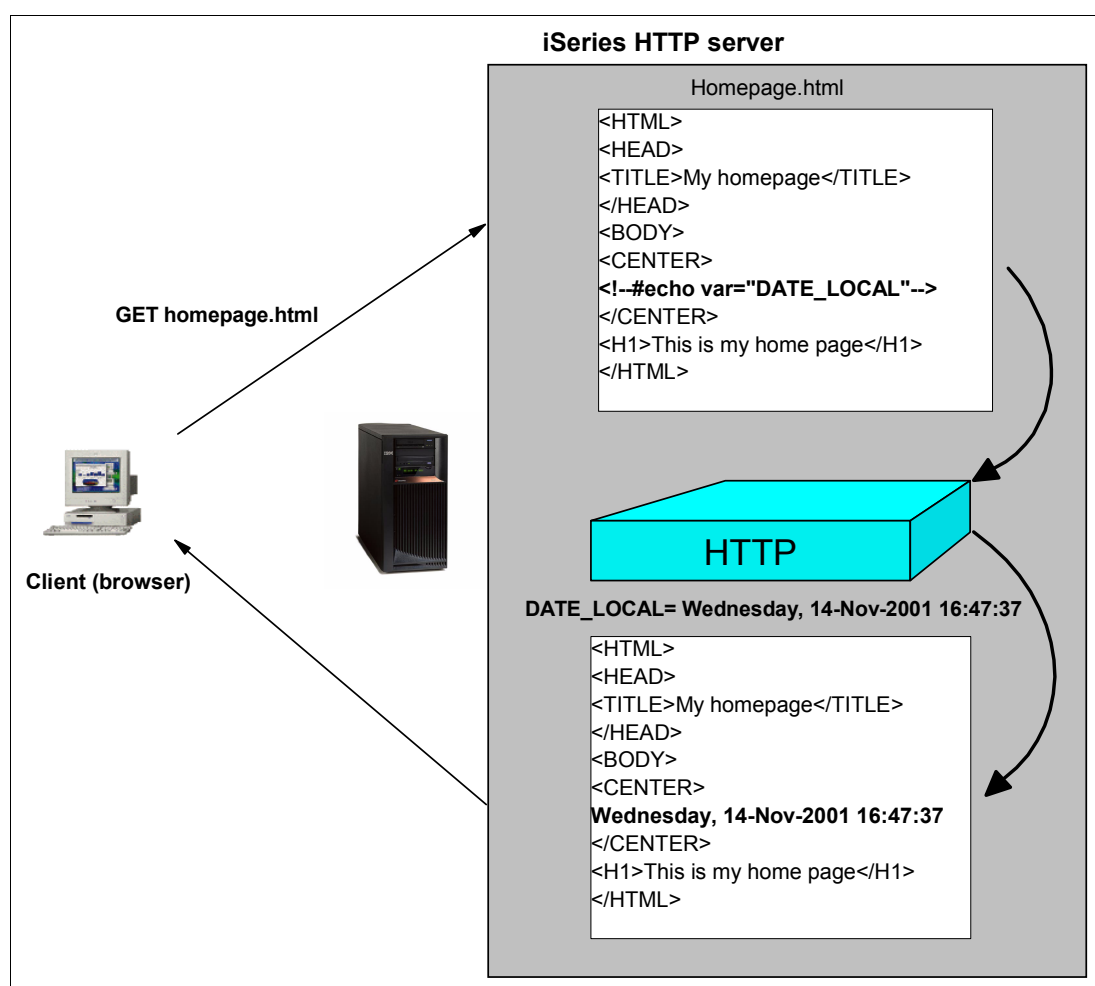


Figure 7-1 SSI processing

SSI also supports the execution of simple conditional statements. Therefore, it provides a reasonably flexible programming environment. The syntax used for SSI directives is:

```
<--#command parameter="value" -->
```

This syntax allows SSI directives to remain hidden when the server is not configured for SSI support.

Tip: Special characters inside SSI directives *must* be preceded by a backslash (\).

Table 7-1 lists the SSI commands available on iSeries and their respective parameters. For additional information, see the Reference section of the HTTP Documentation Center at:

<http://www.ibm.com/eserver/iseries/software/http/docs/doc.htm>

Table 7-1 SSI commands

Command	Description	Valid parameters
config	Controls various output formats.	errmsg, sizefmt, timefmt
echo	Prints one of the SSI or application programming interface (API) variables. Dates are printed using config timefmt.	var, encoding
exec	Calls a CGI program.	cgi
fsize	Prints the size of the specified file according to config sizefmt.	file, virtual
lastmod	Prints the last modification date of the specified file according to config timefmt.	file, virtual
global	Same as the set command.	var, value
include	Inserts the text of another file. Included files can be nested.	(file path)
printenv	Prints all existing environment variables and their values. There are no attributes.	(var name)
set	Sets the value of an environment variable.	var, value

Implementation

The implementation requires the following steps as shown in Figure 7-2:

1. From the Server area list, select the context for which SSIs will be enabled. In our example, we select the root directory (/).

Note: Each HTML file that is served or configured for SSI must be scanned by the HTTP Server (powered by Apache). Only configure SSI in those directories where the SSIs will be used.

2. In the left pane, under Server Properties, select **Dynamic Content and CGI**.
3. Select the **Server Side Includes** tab.

4. On the Server Side Includes page (Figure 7-2), complete these tasks:
 - a. Select either **Allow server-side files without CGI** or **Allow server-side files with CGI program calls inside**.
 In this example, only SSI include directives in files with the extension .shtml are parsed and processed. If, for example, SSI directives are also processed for files with an extension of .html, you must add the file extension to the corresponding table on the Server Side Includes tab.
 - b. Click **Apply**.
 - c. Click **OK**.

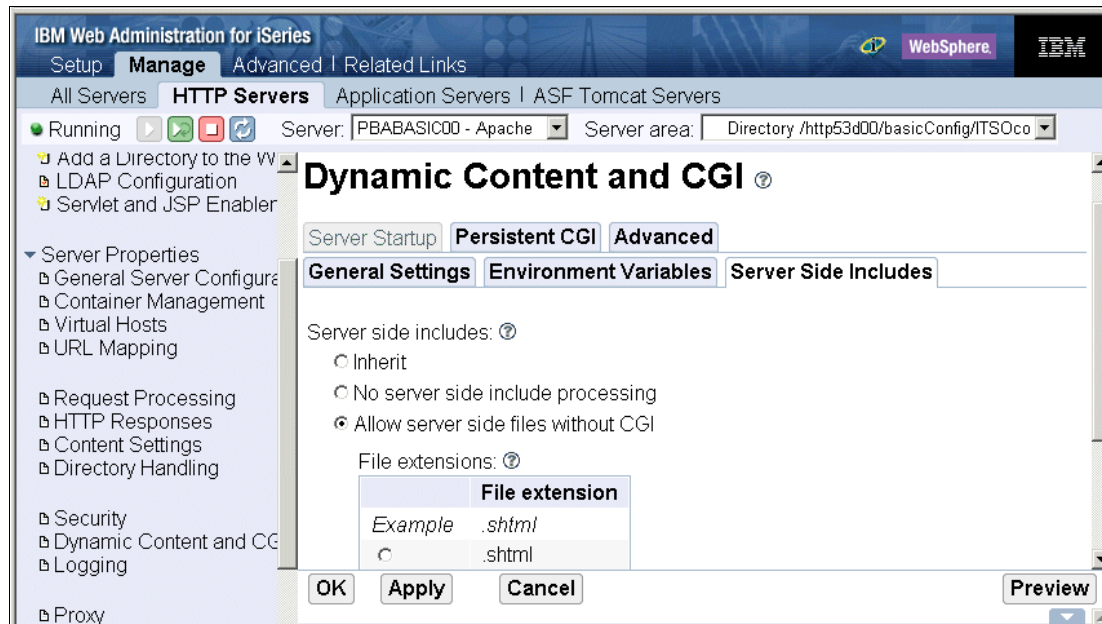


Figure 7-2 Dynamic Content and CGI: Enabling SSI

5. Restart your server.

To test SSI, add a simple directive like the following example to one of your HTML files:

```
<center><!--#echo var="DATE_LOCAL" --></center>
```

The server parses the directive and replaces the SSI command with the value of the environment variable.

7.2 Everything dynamic with CGI support

CGI is a set of programming specifications used to design programs that produce dynamic content. CGI programs process user input submitted through a POST or GET method, returning output to the browser window. CGI programs for the HTTP Server (powered by Apache) can be written in C++, REXX, ILE C, ILE RPG, or ILE COBOL.

Note: Java CGI is no longer supported with IBM HTTP Server for iSeries. Java CGI was supported on releases prior to V5R3 with Java Developer Toolkit (JDK) Version 1.1 only. There is no support for JDK levels other than 1.1. Since JDK 1.1 is no longer shipped with the 5722-JV1 product in V5R3, Java CGI is no longer supported with IBM HTTP Server (5722-DG1) in this release.

The iSeries server can even run a CGI application that is written and compiled for AIX®. The binary output of the compiler is executed directly from OS/400 Portable Application Solutions Environment (OS/400 PASE). For a detailed example of how to implement an AIX binary as a CGI, see Appendix A, “Bringing PHP to your iSeries server” on page 387.

CGIs come into play whenever a significant processing load must be employed to generate dynamic output.

Implementation

The best implementation guide for CGI programming both with the HTTP Server (original) and HTTP Server (powered by Apache) is *HTTP Server for iSeries Programming*, GC41-5435.

For an example of how to configure your HTTP Server (powered by Apache) for a CGI application, see 7.3, “Net.Data: A ready-made scripting tool” on page 161. For other resources and Web sites, see 7.4, “For more information” on page 169.

7.3 Net.Data: A ready-made scripting tool

Net.Data is an easy-to-use scripting language developed by IBM and bundled with the IBM HTTP server (as shown in Table 2-2 on page 20). Net.Data macros are fed to a CGI interpreter (DB2WWW.PGM) that generates HTML output. The Net.Data macro language allows you to generate a wide range of dynamic content through embedded dynamic Structured Query Language (SQL) invocations and program calls. A single HTTP Server (powered by Apache) can handle many Net.Data macros and applications.

7.3.1 Implementation: Setting up the Net.Data environment

A Net.Data environment consists of the macro processor, a configuration file (known as the initialization or INI file), the Net.Data macro source script, and an HTTP server. This example assumes the library and integrated file system (IFS) structure as shown in Figure 7-3.

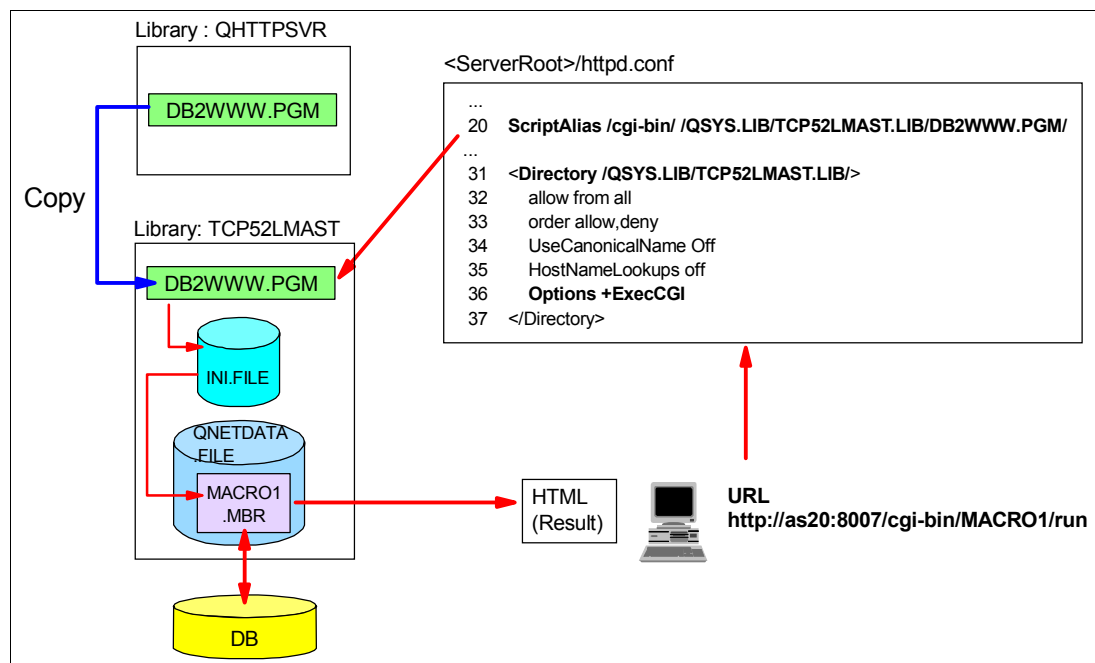


Figure 7-3 Net.Data: Overview of the configuration objects

Table 7-2 defines the main configuration features that you can use to configure an HTTP Server (powered by Apache) to use a Net.Data macro to create dynamic content.

Table 7-2 HTTP Server (powered by Apache) and Net.Data configuration parameters

Parameter	Value
Server name	PBABASIC07
Server root	/tcp52dmast/basicConfig
Document root	/tcp52dmast/basicConfig/ITSOco
IP address	All
Port	8007
OS/400 Library	TCP52LMAST
Maco processor program	DB2WWW.PGM
Directory path for alias	/QSYS.LIB/TCP52LMAST.LIB/DB2WWW.PGM/

To set up the Net.Data environment, follow these steps:

1. Create a user library to have a place to keep these files. Enter the Create Library (CRTLIB) command and create library TCP52LMAST as shown in Figure 7-4.

Note: You may have already restored this library if you followed the steps outlined in Appendix D, “Additional material” on page 421. In this case, you may skip these steps since we created these objects for you.

```

                                Create Library (CRTLIB)

Type choices, press Enter.

Library . . . . . TCP52LMAST      Name
Library type . . . . . *PROD        *PROD, *TEST
ASP number . . . . . 1             1-32, *ASPDEV
ASP device . . . . . *ASP          Name, *ASP, *ASPGRPPRI...
Text 'description' . . . . . iSeries TCP/IP and pbApache

```

Figure 7-4 Net.Data: Creating a new library

2. Copy the original macro processor program DB2WWW.PGM from the QHTTPSVR library to your library, which you just created. Enter the Create Duplicate Object (CRTDUPOBJ) CL command as shown in Figure 7-5. The primary reason you should do this is to move the DB2WWW CGI program into a library that you can protect from both the OS/400 point of view and through HTTP server configuration directives. That is, it is considered a more secure practice to place all your CGI applications, including IBM DB2WWW.PGM, into a single protected library.

Tip: If you apply program temporary fixes (PTFs) for the IBM HTTP Server for iSeries (5722-DG1), recopy the DB2WWW program from QHTTPSVR to your own library. This new copy brings any updated code with it.

Create Duplicate Object (CRTDUPOBJ)		
Type choices, press Enter.		
From object	DB2WWW	Name, generic*, *ALL
From library	QHTTPSVR	Name, *LIBL, *CURLIB
Object type	*PGM	*ALL, *ALRTBL, *AUTL...
+ for more values		
To library	TCP52LMAST	Name, *SAME, *FROMLIB...
New object	*OBJ	Name, *SAME, *OBJ
From ASP device	*	Name, *, *CURASPGRP, *SYSBAS
To ASP device	*ASPDEV	Name, *ASPDEV, *...

Figure 7-5 Net.Data: Copying the DB2WWW.PGM macro processor to your library

3. Create the Net.Data initialization file, which will reside in the same library as the macro processor. Enter the Create Source Physical File (CRTSRCPF) CL command to create file INI with a DB2WWW member, as shown in Figure 7-6. Note that since all the statements in the INI file have to be on single lines, we recommend a record length value of 240.

Create Source Physical File (CRTSRCPF)		
Type choices, press Enter.		
File	INI	Name
Library	TCP52LMAST	Name, *CURLIB
Record length	240	Number
Member, if desired	DB2WWW	Name, *NONE, *FILE
Text 'description'	Net.Data initialization file	

Figure 7-6 Net.Data: Creating the INI initialization file

The initialization file contains your default environment settings such as the path where macros are stored, environment variables, and logging and tracing preferences.

4. Enter the Start PDM (STRPDM) command to modify the INI file as shown in Figure 7-7.

Columns . . . :	1 121	Browse
SEU==>		
FMT **	...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...	
***** Beginning of data *****		
0001.00	MACRO_PATH /QSYS.LIB/TCP52LMAST.LIB/QNETDATA.FILE	
0002.00	DTW_PROCESS_REPORT_ON_ERROR = YES	
0003.00	DTW_SHOWSQL YES	
0004.00	DTW_ERROR_LOG_DIR /TCP52DMAST/LOGS	
0005.00	DTW_ERROR_LOG_LEVEL ERROR	
0006.00	DTW_TRACE_LOG_DIR /TCP52DMAST/LOGS	
0007.00	DTW_TRACE_LOG_LEVEL OFF	
0008.00	DTW_TRACE_MERGE_RECORDS NO	
***** End of data *****		

Figure 7-7 Net.Data: INI file

See the *Net.Data Administration and Programming Guide for OS/400* and *Net.Data Reference* manuals on the Web for more information about environment settings:

<http://www-1.ibm.com/servers/eserver/series/software/netdata/docs/doc.htm>

5. Macros can be stored on the iSeries server as members of a source physical file in a library or as stream files (usually with the .d2w or .ndm extension) inside the IFS. The two solutions are equivalent. Choose one and change the MACRO_PATH statement in your configuration file shown in Figure 7-7 to reflect your choice. Figure 7-8 lists the code for the sample macro used in this example.

```

Columns . . . : 1 121                                Edit
SEU==>
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
***** Beginning of data *****
0001.00 %{----- SAMPLE MACRO -----}%}
0002.00 %define{
0003.00 DATABASE="*LOCAL"
0004.00 DTW_HTML_TABLE="YES"
0005.00 %}
0006.00
0007.00 %function(DTW_SQL) MyQuery() {
0008.00 SELECT * FROM TCP52LMAST.ITEM
0009.00 %}
0010.00
0011.00 %HTML(run){
0012.00 <html>
0013.00 <head>
0014.00 <title>Sample macro</title>
0015.00 </head>
0016.00 <body>
0017.00 <center>
0018.00 <h1>Query Results</h1>
0019.00 @MyQuery()
0020.00 </center>
0021.00 </body>
0022.00 </html>
0023.00 %}
***** End of data *****

```

Figure 7-8 Net.Data: A sample macro

This macro establishes a connection to the local database defined in the Work with Relational Database Directory Entries (WRKRDBDIRE) display. When the %html (run) block is called, this connection is used to perform an SQL query on a parts database file, and the output is returned as a basic HTML table inside a dynamically generated page.

7.3.2 Configuring your HTTP Server (powered by Apache) for CGI

This exercise explains how to enable your HTTP Server (powered by Apache) to support dynamically generated Web pages. Table 7-2 on page 162 shows the HTTP Server (powered by Apache) configuration used in this task.

First, you need to create an alias to the library on the iSeries server which contains the CGI program for Net.Data. This alias is used in the Web client Uniform Resource Locator (URL). Therefore, the library structure and physical names of directories and files are not revealed to end users of your Web site.

1. From the Server list, select your server name. From the Server area list, select **Global Configuration** as shown in Figure 7-9.
2. In the left pane, under Server Properties, click **URL Mapping**.
3. In the right panel, select the **Aliases** tab.

4. On the Aliases tab (Figure 7-9), complete these tasks:
 - a. Click **Add** to enter a new URL to host file system mapping.
 - a. Under Alias type, select **Script Alias**.
 - b. Under URL path, type `/cgi-bin/`.
 - c. Under Host directory or file, type `/QSYS.LIB/TCP52LMAST.LIB/DB2WWW.PGM/`.

Attention: Be careful! The URL path and directory or file names are case sensitive in this situation.

- d. Click **Continue**.
- e. Click **OK**.

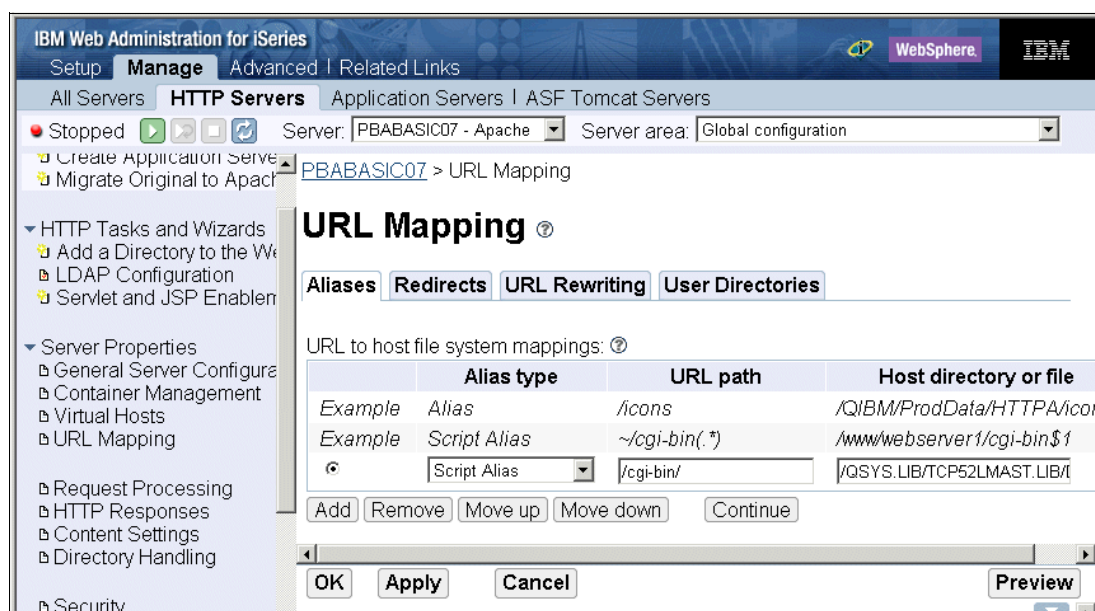


Figure 7-9 Net.Data: Creating the ScriptAlias directive

Next, as shown in Figure 7-10, you tell your server that it is allowed to run CGI programs from this directory. One way to do this is to create a new context (container) for this directory in which you will place directives allowing access to the CGI program (DB2WWW.PGM).

1. In the left pane, under Server Properties, click **Container Management**.
2. In the right panel, select the **Directories** tab.

3. On the Directories page (Figure 7-10), complete these tasks:
 - a. Under Directory/Directory Match containers, you see two entries in the table. The first was created automatically when the server was created using the create wizard. The root directory / is secured by default. The second entry was added by the create wizard. It allows the server to serve the home page, other public Web pages and image files.

Click **Add** to add a new entry to the table.
 - b. Under the Type column, select **Directory**. For the Directory path or expression, type /QSYS.LIB/TCP52LMAST.LIB/. This is the physical path of the library containing the CGI programs. In this case, it is the Net.Data program DB2WWW.PGM as provided by IBM.
 - c. Click **Continue**.
 - d. Click **OK**.

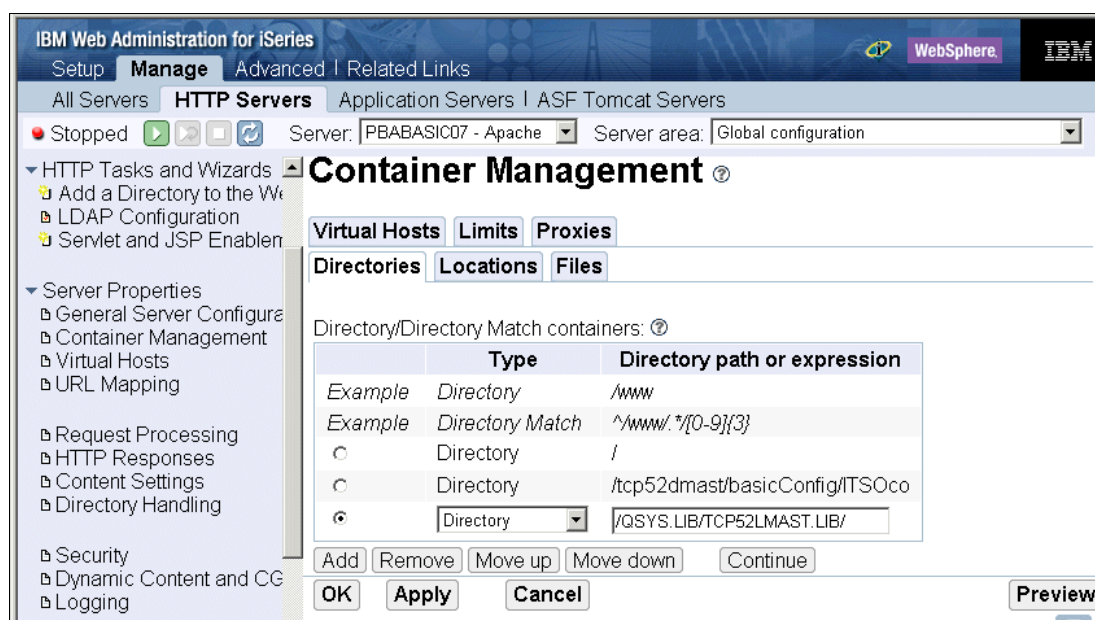


Figure 7-10 Net.Data: Creating a new Directory context (or container)

Next you need to tell the server who is allowed to access the CGI programs. Refer to Figure 7-11 for the following steps.

1. From the Server area list, select the newly created **Directory /QSYS.LIB/TCP52LMAST.LIB/**.
2. In the left pane, click **Security**
3. In the right panel, select the **Control Access** tab.

4. On the Control Access page (Figure 7-11), complete these steps:
 - a. Under Control access based on where the request is coming from, select **Allow then deny** for Order for evaluating access.
 - b. Select **Allow access to all, except the following**.
 - c. Click **Apply**.
 - d. Click **OK**.



Figure 7-11 Net.Data: Control Access

Next you need to tell your server that it is allowed to run CGI programs in this directory. The following steps explain how to do this:

1. In the Server area, make sure that the directory you are working with is still selected.
2. In the left pane, select **Dynamic Content and CGI**.
3. Select the **General Settings** tab.

- On the General Settings page (Figure 7-12), for Allow CGI programs to be run, select **Enabled**.

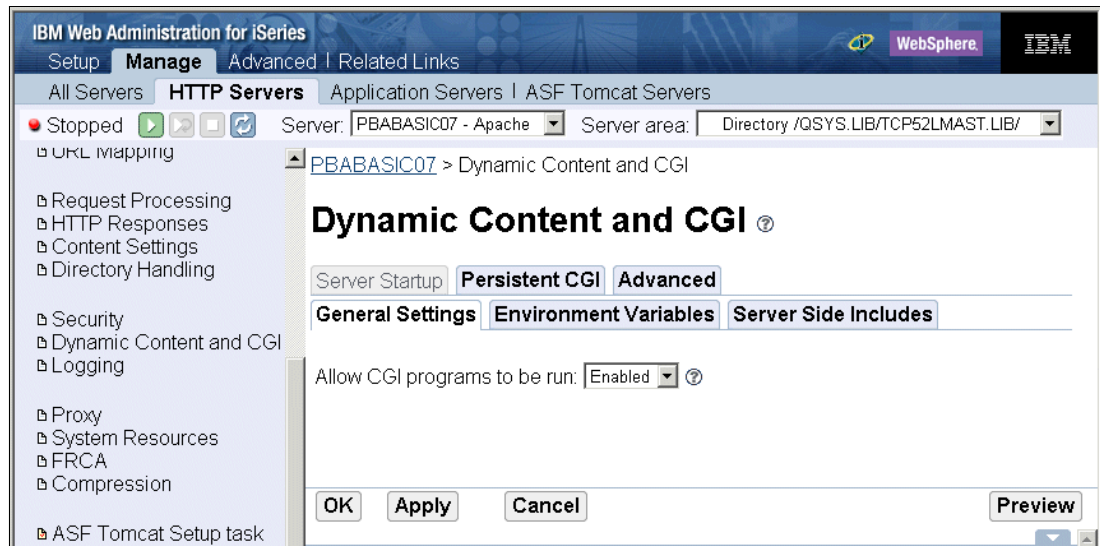


Figure 7-12 Net.Data: Enabling CGI programs to run

- Click **Apply** and then click **OK**.

When displaying your configuration, you should now see your configuration file and its directives as shown in Figure 7-13.

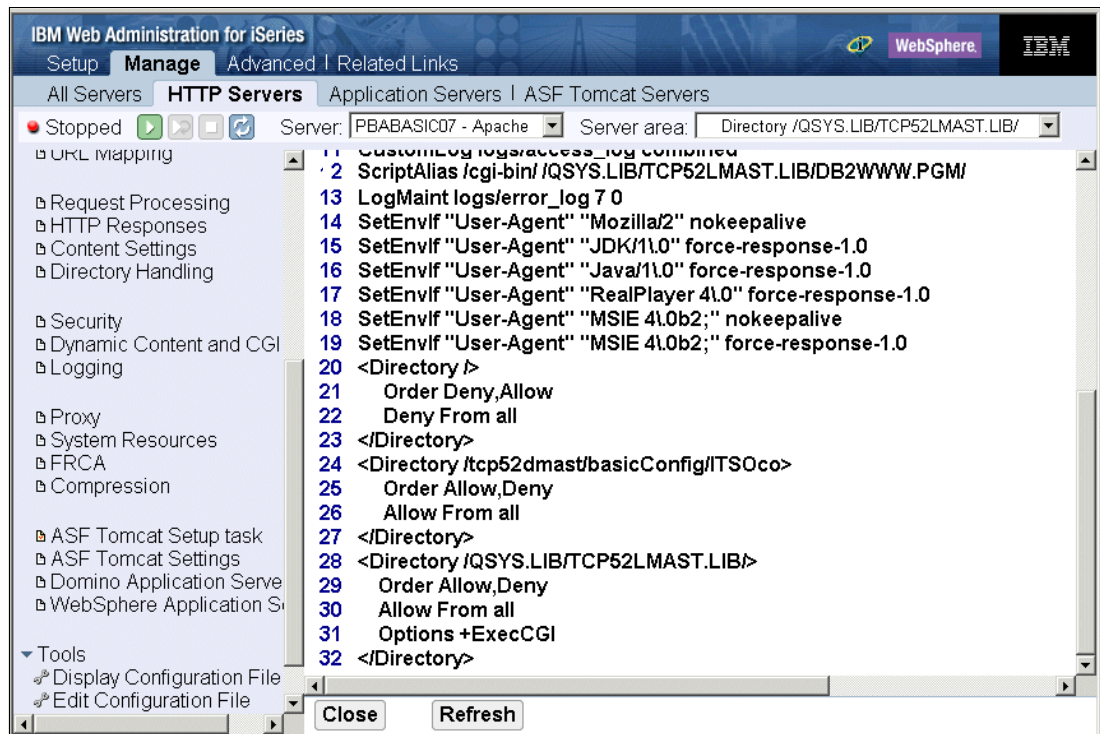


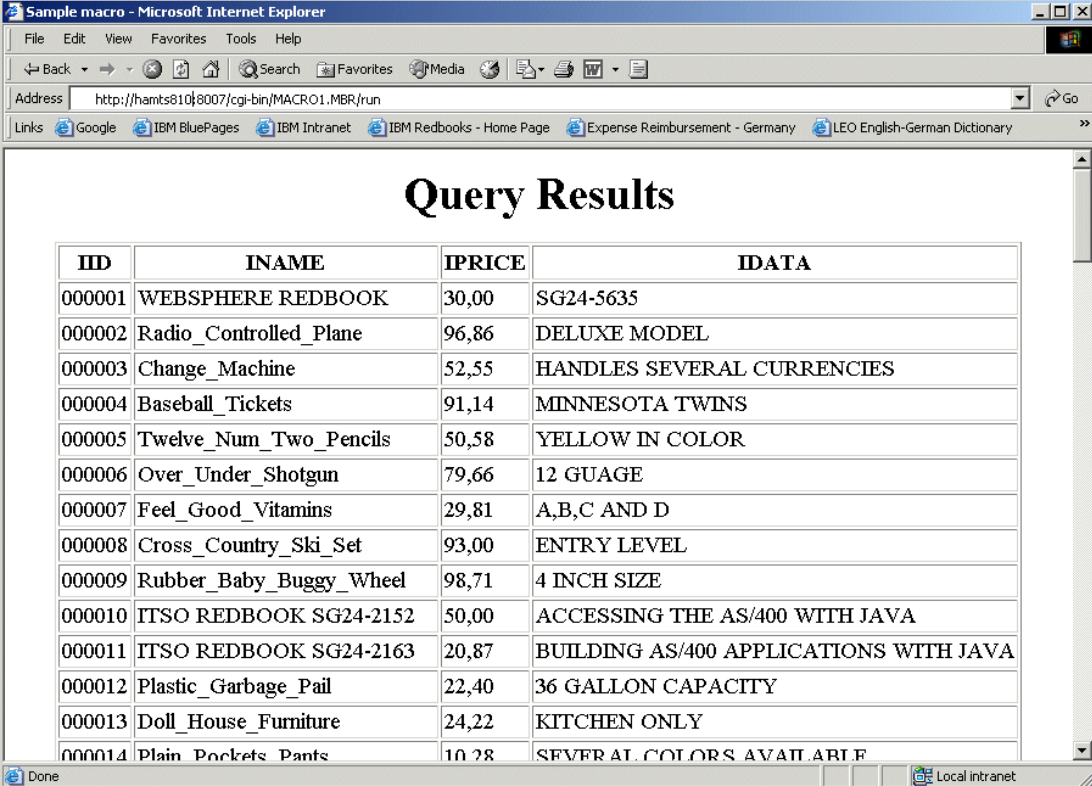
Figure 7-13 Net.Data: Directives for enabling CGI and Net.Data

Your server is now set up to support dynamically generated Web pages via a CGI.

7.3.3 Testing your HTTP Server (powered by Apache) and Net.Data macro

You must stop and start your server to re-read the configuration file. Test the HTTP server by entering the following URL in your Web client to see the results as shown in Figure 7-14:

`http://hamts810:8007/cgi-bin/MACRO1.MBR/run`



IID	INAME	IPRICE	IDATA
000001	WEBSPHERE REDBOOK	30,00	SG24-5635
000002	Radio_Controlled_Plane	96,86	DELUXE MODEL
000003	Change_Machine	52,55	HANDLES SEVERAL CURRENCIES
000004	Baseball_Tickets	91,14	MINNESOTA TWINS
000005	Twelve_Num_Two_Pencils	50,58	YELLOW IN COLOR
000006	Over_Under_Shotgun	79,66	12 GUAGE
000007	Feel_Good_Vitamins	29,81	A,B,C AND D
000008	Cross_Country_Ski_Set	93,00	ENTRY LEVEL
000009	Rubber_Baby_Buggy_Wheel	98,71	4 INCH SIZE
000010	ITSO REDBOOK SG24-2152	50,00	ACCESSING THE AS/400 WITH JAVA
000011	ITSO REDBOOK SG24-2163	20,87	BUILDING AS/400 APPLICATIONS WITH JAVA
000012	Plastic_Garbage_Pail	22,40	36 GALLON CAPACITY
000013	Doll_House_Furniture	24,22	KITCHEN ONLY
000014	Plain Pockets Pants	10,28	SEVERAL COLORS AVAILABLE

Figure 7-14 Net.Data: HTML query results of an iSeries database table created by Net.Data

7.4 For more information

Refer to the following resources for more information:

- ▶ Net.Data manuals and documentation home page
<http://www.ibm.com/eserver/iseries/software/netdata/docs/doc.htm>
- ▶ Sample CGI programs
<http://www.ibm.com/eserver/iseries/software/http/examples/>
- ▶ Easy400, CGI Web development tools Web site for iSeries
This site includes a link to download the CGIDEV2 ILE-RPG CGI Development Toolkit.
<http://www-922.ibm.com/easy400p/easy400p01.html>
- ▶ IGNITE/400 iSeries On Demand Business user group offers sample programs and tips
<http://www.ignite400.org>
- ▶ *Who Knew You Could Do That with RPG IV? A Sorcerer's Guide to System Access and More*, SG24-5402
This IBM Redbook has an example CGI application written in RPG IV.



Part 3

Building a Web application

If you started reading this IBM Redbook from the beginning to this point, you should now have a good fundamental understanding of the HTTP Server (powered by Apache). And If you have made it this far, we know now that you are serious about doing more than just providing a Web site with a bit of dynamic data. You are serious about building a Web application based upon an On Demand Business infrastructure. In this case, the following chapters were written for you.

This part takes an in-depth look at the HTTP Server (powered by Apache). It includes:

- ▶ The steps that are necessary to implement Web application serving with Java featuring WebSphere Application Server and the Apache Software Foundation's (ASF) Tomcat.
- ▶ A comparison guide of the strengths and weaknesses of both Web application servers.
- ▶ Advanced topics such as how to get the best performance from your HTTP Server (powered by Apache), an introduction to the Webserver Search Engine, problem determination, high availability, and national language considerations.
- ▶ A running example of extending the core features of your HTTP Server (powered by Apache) via Apache Portable Runtime (APR) support, which allows you to write your own modules or port them to the iSeries as Integrated Language Environment (ILE) service programs.

Indeed, if you are serious about your HTTP server and the service that it provides for your customers, then you may be interested in these advanced resources too:

- ▶ Apache Software Foundation is the “galactic center” of all documentation and information about the Apache server and all other related projects such as Jakarta Tomcat, Hypertext Preprocessor (PHP), and so on. To learn more, see:

<http://www.apache.org>

- ▶ The IBM Systems Group Services' IBM Custom Technology Center is an organization of more than 70 world class application architects and programmers (one is located in Rochester, Minnesota, home of your iSeries) and is an extension of the IBM development laboratories. They specialize in rapid design and architecture, infrastructure development, software installation and configuration, development of applications across multiple technologies, and IBM TotalStorage® services. They have worked with over 4000 customers and have more than 400,000 hours of services engagements over the past ten years. You can find more information on the Web at:

<http://www.ibm.com/servers/eserver/services/>

- ▶ IBM developerWorks® has an extensive collection of programming tips and sample applications, white papers, education, and pointers to IBM alphaWorks® and IBM PartnerWorld®. A search of this site for the keyword Apache yielded 493 results. For more information, see:

<http://www.ibm.com/developerworks>

- ▶ IBM alphaWorks is a place for programmers to meet and share leading-edge applications and information.

<http://www.alphaworks.ibm.com/>

- ▶ IBM PartnerWorld is a collection of resources for IBM Business Partners.

<http://www.ibm.com/partnerworld>

- ▶ Many excellent third-party Web sites focus on different aspects of the iSeries server. Here is a partial list.

- Search400.com

<http://search400.techtarget.com/>

- IGNITE/400

<http://www.ignite400.org/>

- COMMON

<http://www.common.org/>

- Midrange Computing Press Online

<http://www.mcpressonline.com/>

- iSeries Network

<http://www.iseriesnetwork.com/>

- @server Magazine, iSeries edition (formerly *iSeries Magazine*), which is an IBM publication

<http://eservercomputing.com/iseries/>



Migration from HTTP Server (original) to (powered by Apache)

Some HTTP servers running in the iSeries server are the HTTP Server (original). This server has been supported on the iSeries server since V4R3 of OS/400. However, with V4R5, a new Web server came to the iSeries server and customers had the option to choose between the HTTP Server (original) and the HTTP Server (powered by Apache). Now, with i5/OS V5R3, the only supported Web server is the HTTP Server (powered by Apache). This also means that your HTTP Server (original) instance will no longer run under i5/OS.

Both servers have similar functions and can coexist and run together on the same iSeries server. After all, an HTTP server is nothing more than a fancy file server. We use configuration directives to tell the server which files to serve and which ones to protect. However, they have some differences. This chapter does not detail these differences (see 1.1, “HTTP Server (powered by Apache) features” on page 4). Instead, it defines them well enough so that you can understand the strengths and limitations of migrating a configuration from the HTTP Server (original) to the HTTP Server (powered by Apache).

The HTTP Server (original) evolved from work being done at the European Laboratory for Particle Physics (CERN) in Geneva, Switzerland. The Apache server was developed at the National Center for Supercomputing Application (NCSA) and was based on the NCSA HTTP daemon (NCSA HTTPd 1.3). If you are familiar to the HTTP Server (original) configuration, you will find Apache’s configuration a bit different.

Tip: The total move to Apache with i5/OS V5R3 didn't come by surprise. It has been the goal of IBM to eventually replace the HTTP Server (original) with the HTTP Server (powered by Apache) since 07 March 2003, when the following announcement was made, which you can find on the Web at:

<http://www.ibm.com/servers/eserver/iseries/software/http/news/sitenews.html>

V5R2 to be the final release to support HTTP Server (original)

IBM plans for V5R2 to be the final release to support HTTP Server (original). IBM HTTP Server (powered by Apache) is the recommended solution for your Web serving needs. IBM plans for the HTTP Server (original) to be removed from IBM HTTP Server for iSeries in a future release. For more information about migrating HTTP Server (original) configurations to HTTP Server (powered by Apache), visit our migration article:

<http://www.ibm.com/servers/eserver/iseries/software/http/product/migrate.html>

For long-term thinkers, HTTP Server (powered by Apache) is the better choice.

8.1 A look at HTTP Server (original) and (powered by Apache)

If you have been serving your domain using another Web server and you want to use the HTTP Server (powered by Apache), you have three options:

- ▶ Create a new HTTP Server (powered by Apache) configuration from scratch
- ▶ Migrate the HTTP Server (original) configuration
- ▶ Port an Apache configuration from a non-iSeries server

This option, while valid, is not discussed in this chapter. Almost all directives migrate smoothly to the iSeries server. These are some of the places you should first look to identify if there is a possible problem:

- The configuration file should be based on Apache Version 2.0. Many other platforms are still at Version 1.3.
- The syntax of the LoadModule directive will most likely be different. At the very least, modules on your iSeries server are provided by Integrated Language Environment (ILE) service programs and generally look like this:

```
LoadModule header_module /QSYS.LIB/ITSOAPACHE.LIB/MOD_HEADER.SRVPGM
```

- This also brings up another point that many people have invested a lot of time and effort to extend the capabilities of their Apache server running on a specific platform. You must look for those platform-specific extensions to the Apache server and either port them to the iSeries or find an equivalent replacement.

An example of this may be Perl or Hypertext Preprocessor (PHP) scripts. See Appendix A, "Bringing PHP to your iSeries server" on page 387, for information about PHP. For more information about Perl for your iSeries, go to:

- <http://www.iseries.ibm.com/tstudio/workshop/tiptools/perl.htm>
- <http://www.cpan.org/ports/index.html#os400/>

Regardless of which path brings you to the HTTP Server (powered by Apache), in the end, of course, you must test.

The HTTP Server (original) can be migrated using the migration wizard provided with the IBM HTTP Server for iSeries. The success of migrating the HTTP Server (original) configuration file depends on the server directives used in the original configuration file. Most directives can be migrated cleanly. Few directives are not migrated because they are no longer supported. Maybe the best way to think about the migration wizard is that it is a tool to help you. Like any tool, it is your ultimate responsibility to end up with a clean migration.

You will notice that for either option (a new HTTP Server (powered by Apache) configuration or a migration of the HTTP Server (original)), you must *test thoroughly* the new HTTP configuration file. You can perform the test without any HTTP Server (original) interruption because the migration utility does not remove or modify the HTTP Server (original) configuration file. Both servers, the original and migrated, can be active simultaneously. However they must listen on different ports or Internet Protocol (IP) addresses.

One of the most important differences between the two servers is the Apache concept of context. All server directives are related to a context and have meaning only within that context. Some contexts are:

- ▶ General settings
- ▶ VirtualHost
- ▶ Location
- ▶ Directory

You must understand the Apache contexts to understand the output of the migration tool. Refer to Chapter 4, “Quick guide to Apache contexts and request routing” on page 59, for additional information.

There are also differences related to the server directives supported by each HTTP server.

8.1.1 Directives and services not supported

Some directives are no longer supported. The HTTP Server (powered by Apache) does not support:

- ▶ Log reporting and Web usage mining
However, the Real Time Server Statistics that are now available with the HTTP Server (powered by Apache) provide a similar function.
- ▶ Platform for Internet Content Selection (PICS) support
- ▶ The Server API as provided with the HTTP Server (original). The HTTP Server (powered by Apache) uses Apache Portable Runtime (APR) application programming interfaces (APIs). See Chapter 12, “Apache Portable Runtime: Extending your core functionality” on page 311.
- ▶ The NameTrans directive used by WebSphere Application Server (which is also not needed)

In addition, the PUT and DELETE methods used by the HTTP Server (original) require WebDAV configuration on the HTTP Server (powered by Apache).

8.1.2 Equivalent directives

Most of the HTTP Server (original) directives have an equivalent directive in the HTTP Server (powered by Apache) and the migration utility migrates those directives cleanly. However some functions require additional configurations. Table 8-1 lists the most commonly used original directives and their Apache equivalent.

Table 8-1 Mapping HTTP Server (original) to HTTP Server (powered by Apache) directives

HTTP Server (original) directives	HTTP Server (powered by Apache) directives
Pass	Alias and AliasMatch. Note: The migration wizard mostly uses AliasMatch as the first choice in migrating a Pass directive.
Exec	ScriptAlias and ScriptAliasMatch Note: The migration wizard mostly uses ScriptAliasMatch as the first choice in migrating an Exec directive.
Map	MapMatch
Fail	Deny in <location>

8.1.3 Functional differences

The HTTP Server (powered by Apache) has some functional differences from the HTTP Server (original) in the area of server-side includes (SSI):

- ▶ The HTTP Server (powered by Apache) ignores the **cmntmsg** attribute of the configuration command.
- ▶ The HTTP Server (powered by Apache) does not allow extra text after a tag value.

8.1.4 New HTTP Server (powered by Apache) directives

The HTTP Server (powered by Apache) includes a set of new server directives. That is, you and the migration tool can take advantage of these directives to enhance the functionality, performance, and scalability of any migrated HTTP Server (original) instance. Table 8-2 describes some of these new server directives.

Table 8-2 New HTTP Server (powered by Apache) directives

Apache directive	Description	Purpose
AccessFileName	Specifies name of ACL file	Security
AllowOverride	ACL values overriding others	Security
<Directory>	Defines attributes for directories	Security
<DirectoryMatch>	Same, but uses regular expression	Server configuration and virtual host
DocumentRoot	Defines the default directory from which all Web content must be served from	Server configuration and virtual host
<Files>	Defines attributes for files	Security
<FilesMatch>	Same as above	Security
<Limit>	Group access control based on method	All

Apache directive	Description	Purpose
<LimitExcept>	Opposite of <Limit>	All
LoadModule	Enables modules compiled in but not in use	Support new functions
MapMatch	To migrate Map directives from HTTP Server (original)	Uniform Resource Locator (URL) mapping
NameVirtualHost	Specifies address for name-based virtual host names	Virtual hosts
ServerAlias	Virtual host names	Virtual host
ServerRoot	Directory where server is installed	General setting
UseCanonicalName	Allows use of short names	Dynamic virtual host
<VirtualHost>	Group directives based on host	Virtual host

8.2 An example migration

If you want to migrate an HTTP Server (original) instance, you can do it using the migration wizard. This wizard migrates the server directives found in your HTTP Server (original) but without adding any new functionality. You can use this migration utility and leave the migrated server as it was migrated or you can enhance it using new server directives.

The migration process includes the following steps:

1. Understand the initial situation of the HTTP Server (original) configuration.

Tip: Due to the differences in the way the HTTP Server (powered by Apache) server processes the configuration file and directives compared to the HTTP Server (original), you should clean your source configuration file before migration. For example, when the HTTP Server (original) processes the configuration file, it stops at the first match. If you have a directive that is less specific (Pass /* /webdocs/default/*) than another one later in the file (Pass /help/* /webdocs/help/*), the second one is never processed. With the HTTP Server (powered by Apache) server, both Pass statements are migrated and processed. This can lead to unwanted results.

2. Follow the migration steps as explained in 8.2.2, “Migration steps” on page 178.
3. Test your migration as explained in 8.3, “Testing your migration” on page 188.
4. Fix any bugs (see Chapter 13, “Problem determination: When things do not go as planned” on page 323).
5. Repeat these steps for each directive until you are finished or create a brand new HTTP Server (powered by Apache) configuration.

Testing is one of the most important tasks when migrating from an HTTP Server (original). Without testing, you cannot ensure the functionality of the migrated server.

You can perform testing without any HTTP Server (original) interruption because the migration utility does not remove or modify the HTTP Server (original) configuration file. Both servers, the original and migrated, can be active simultaneously under OS/400 V5R2. However, you must run them in different ports or IP addresses. Under i5/OS V5R3, the HTTP Server (original) no longer starts.

8.2.1 Initial situation: HTTP Server (original) configuration

To show the migration process flow, we selected the DEFAULT server defined by the CONFIG configuration file (Example 8-1).

Example 8-1 HTTP Server (original) configuration

```
Pass / /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/Welcome.html
Pass /sample/* /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/*
Pass /QIBM/ProdData/OS400/SQL/Samples/
IconPath /QIBM/HTTPSVR/Icons/
AddIcon text.gif      text  text/*
AddIcon html.gif      html  text/html
AddIcon binary.gif    bin   application/*
AddIcon compress.gif  Z     application/x-compress
AddIcon compress.gif  gzip  application/x-gzip
AddIcon image.gif     img   image/*
AddIcon movie.gif     vid   video/*
AddIcon sound.gif     au    audio/*
AddType .java text/plain binary 1.0
AddType .html text/html 8bit 1.0
AddType .htm text/html 8bit 1.0
AddType .gif image/gif  binary
AddType .bmp image/bmp  binary 1.0
```

8.2.2 Migration steps

To migrate the HTTP Server (original) to an HTTP Server (powered by Apache), follow these steps:

1. Click the **Setup, Manage, or Advanced** tab.
2. In the left pane, under Common Tasks and Wizards, click **Migrate Original Server to Apache**.
3. In the right panel, the Migrate Original to Apache wizard opens (see Figure 8-1). Complete these tasks:
 - a. Select the original server that you want to migrate by either choosing the **HTTP server (original)** or **Named configuration** option.
 - b. Select the appropriate server name or configuration name from the list. For this example, we selected **CONFIG** which is the name of the configuration file for the DEFAULT server.

Tip: The main difference between the two options for selecting a configuration is that the first option shows only HTTP instances that are defined on this system. That is, you see the instances in the IBM Web Administration for iSeries interface that are configured on a system prior to V5R3. Starting from V5R3, instances do not exist anymore. Only configuration files are available for migration. The Named configuration option allows you to select configuration members that are stored in library QUSRSYS in the file QATMHTTPC. The latter option allows you to transfer a configuration member from one server and perform the migration on another server.

- c. Click **Next**.

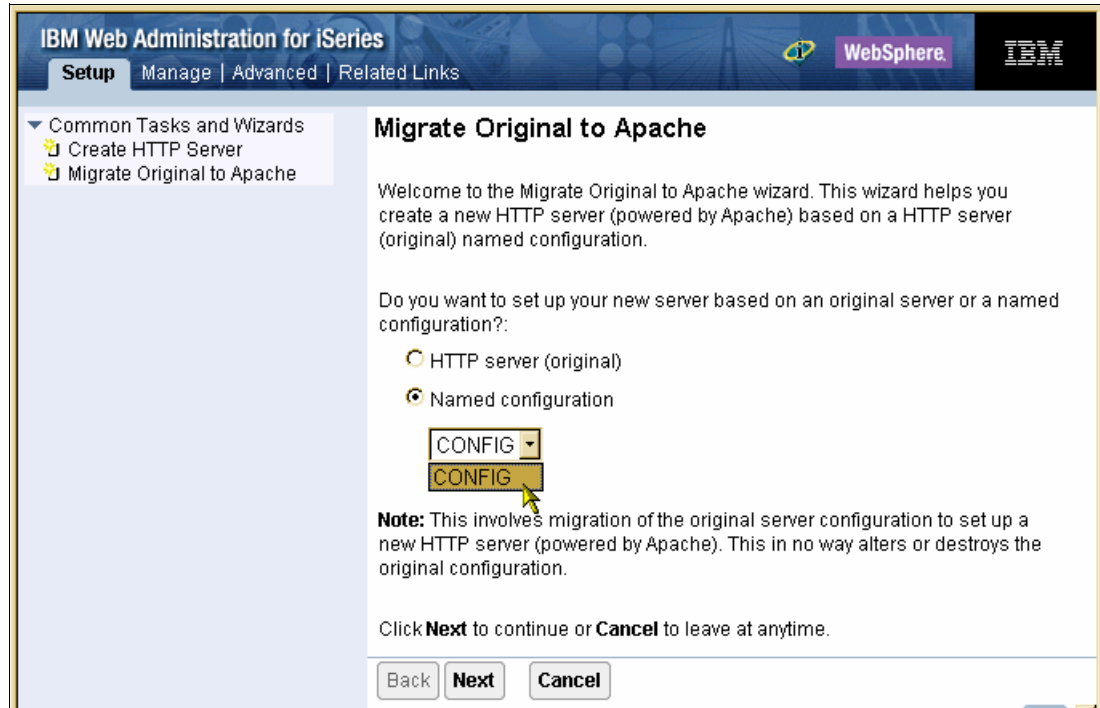


Figure 8-1 Migration: Selecting the original server

- In the next panel (Figure 8-2), specify the new HTTP Server (powered by Apache) name. For this example, we kept the default name WEBSERVER and added a meaningful description. Click **Next**.

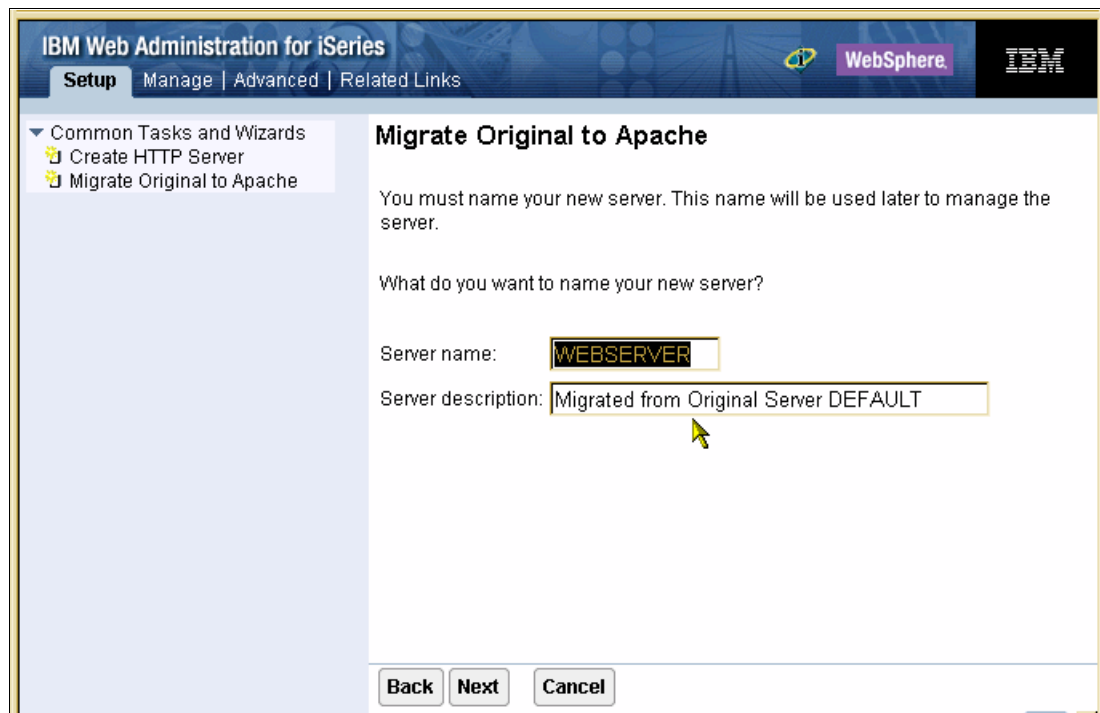


Figure 8-2 Migration: Specifying the new HTTP Server (powered by Apache) name

5. In the next panel (Figure 8-3), enter the server root for the new server. In this example, we used the default /www/webserver. Click **Next**.

IBM Web Administration for iSeries

Setup | Manage | Advanced | Related Links

WebSphere IBM

Common Tasks and Wizards

- Create HTTP Server
- Migrate Original to Apache

Migrate Original to Apache

The server root is the base directory for your server. Within this directory, the wizard will create subdirectories for your logs and configuration information. Supported file systems for the server root are root and QOpenSys.

Which directory would you like to use as the server root for your new server?

Server root:

Figure 8-3 Migration: Specifying the server root for the new server

6. With the HTTP Server (original) identified and the HTTP Server (powered by Apache) attributes defined, it is time to migrate the configuration.

The migration utility mentions some basic differences between both HTTP servers. It also highlights the message that the HTTP Server (original) configuration file is not modified in any way as you can see in Figure 8-4. Click **Next**.

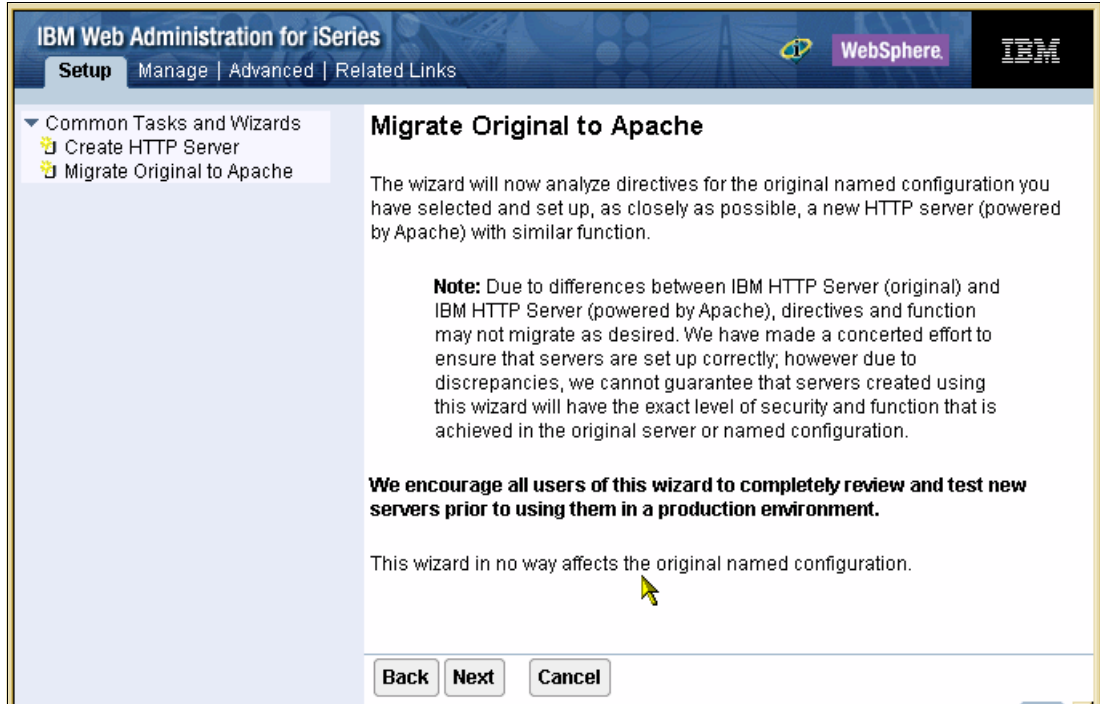


Figure 8-4 Migration: Process

At this point, the migration utility has finished analyzing the original named configuration. It allows you to review a summary of the number of directives that were found in the HTTP Server (original). It also enables you to review the number of Apache directives that were used to provide the same functionality. Figure 8-5 shows an example of this report.

Note: If your HTTP Server (original) configuration contains Map directives, an additional page is shown.

Migrate Original to Apache

The wizard has detected Map directives in the original server configuration which require the use of MapMatch directives for your new server.

The MapMatch directive is supported by HTTP server (powered by Apache) on iSeries for the sole purpose of aiding in the migration of HTTP server (original) configurations. This directive may not be supported on other platforms. Therefore, you are being given the opportunity to select whether it should use the MapMatch directive for this migration. If you choose not to use this directive, you will be required to manually migrate the following directives from the original server configuration: Map · Pass · Exec · Fail · Redirect

Do you want the wizard to migrate request routing directives using the MapMatch directive?

- ☒ Yes - **recommended**
☐ No

This option gives you the choice to have the migration wizard migrate all request routing directives including the map directive. The migrator converts the map directive into a MapMatch directive, which is known only on the iSeries. To use directives that are common to Apache configurations, you may want to select No at the question whether you want to migrate using MapMatch directives. In this case, you have to manually migrate all request routing directives.

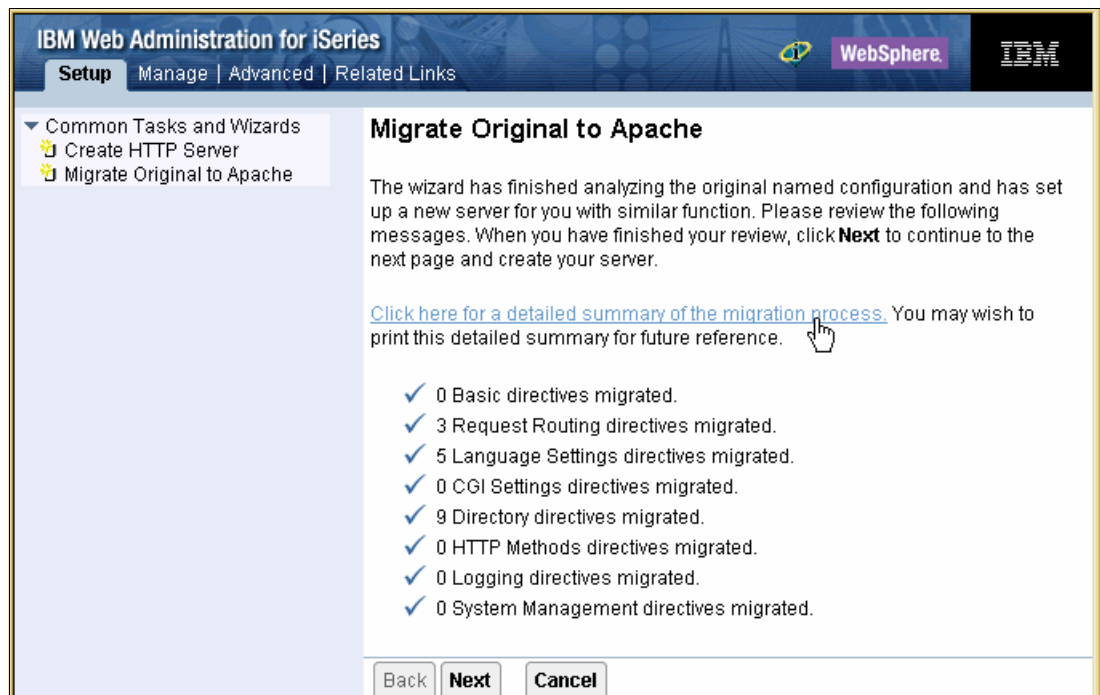


Figure 8-5 Migration: Summary report of migration of original to HTTP Server (powered by Apache)

As indicated in Figure 8-5, you can see the link “Click here for a detailed summary of the migration process.” It takes you to the details behind the success or failure as found by the migration utility. The migration utility defines the success of the migrated server directives within the following categories:

- Basic directives
- Request routing
- Language Settings directives
- CGI directives
- Directory directives
- HTTP methods
- Logging
- System managements
- Directives not migrated

The details of the report may look as shown in Example 8-2.

Example 8-2 Report details

```
0 Basic directives migrated.
HTTP server (original) directives
HTTP server (powered by Apache) directives
Listen *:80
DirectoryIndex welcome.html index.html

-----

3 Request Routing directives migrated.
HTTP server (original) directives
Pass / /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/Welcome.html
Pass /sample/* /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/*
Pass /QIBM/ProdData/OS400/SQL/Samples/
HTTP server (powered by Apache) directives
AliasMatch ^/$ /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/Welcome.html
Directory /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/Welcome.html/
Directory /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/
FilesMatch ^Welcome\.html$
Allow From all
Allow From all
AliasMatch ^/sample/(.*) /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/$1
Allow From all
AliasMatch ^/QIBM/ProdData/OS400/SQL/Samples/$ /QIBM/ProdData/OS400/SQL/Samples/
Directory /QIBM/ProdData/OS400/SQL/Samples/
Allow From all

-----

5 Language Settings directives migrated.
HTTP server (original) directives
AddType .java text/plain binary 1.0
AddType .html text/html 8bit 1.0
AddType .htm text/html 8bit 1.0
AddType .gif image/gif binary
AddType .bmp image/bmp binary 1.0
HTTP server (powered by Apache) directives
AddType text/plain .java
AddType text/html .html
AddType text/html .htm
AddType image/gif .gif
AddType image/bmp .bmp
```

```
-----  
0 CGI Settings directives migrated.  
HTTP server (original) directives  
HTTP server (powered by Apache) directives  
CGIConvMode %MIXED/MIXED%  
-----
```

```
9 Directory directives migrated.  
HTTP server (original) directives  
IconPath /QIBM/HTTPSVR/Icons/  
AddIcon text.gif text text/*  
AddIcon html.gif html text/html  
AddIcon binary.gif bin application/*  
AddIcon compress.gif Z application/x-compress  
AddIcon compress.gif gzip application/x-gzip  
AddIcon image.gif img image/*  
AddIcon movie.gif vid video/*  
AddIcon sound.gif au audio/*  
HTTP server (powered by Apache) directives  
HeaderName README  
AddIconByType "/QIBM/HTTPSVR/Icons/text.gif" text/*  
AddAltByType "text" text/*  
AddIconByType "/QIBM/HTTPSVR/Icons/html.gif" text/html  
AddAltByType "html" text/html  
AddIconByType "/QIBM/HTTPSVR/Icons/binary.gif" application/*  
AddAltByType "bin" application/*  
AddIconByType "/QIBM/HTTPSVR/Icons/compress.gif" application/x-compress  
AddAltByType "Z" application/x-compress  
AddIconByType "/QIBM/HTTPSVR/Icons/compress.gif" application/x-gzip  
AddAltByType "gzip" application/x-gzip  
AddIconByType "/QIBM/HTTPSVR/Icons/image.gif" image/*  
AddAltByType "img" image/*  
AddIconByType "/QIBM/HTTPSVR/Icons/movie.gif" video/*  
AddAltByType "vid" video/*  
AddIconByType "/QIBM/HTTPSVR/Icons/sound.gif" audio/*  
AddAltByType "au" audio/*  
-----
```

The directives not supported by the Apache server, if any, are also included in the migration summary.

Click **Next**.

7. Now you see summary page as shown in Figure 8-6. Click **Finish** to generate the migrated server configuration.

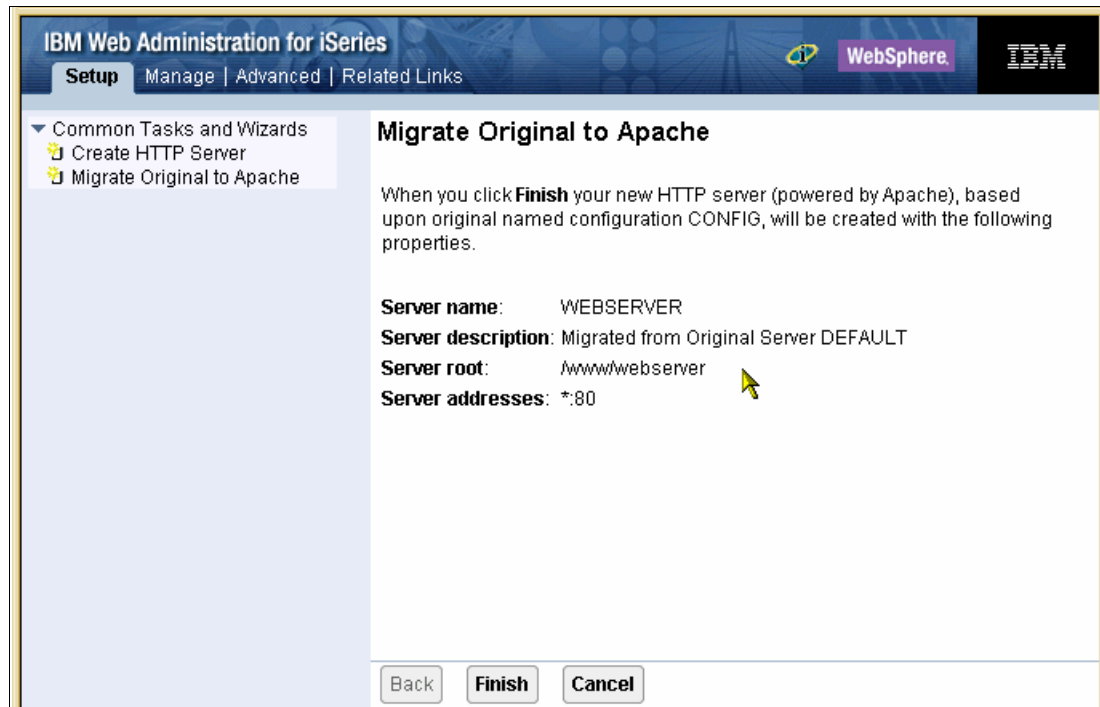


Figure 8-6 Migrate Original to Apache

8. The Manage Apache server "WEBSERVER" panel opens as shown in Figure 8-7. As shown, you can also see the version of Apache installed here.

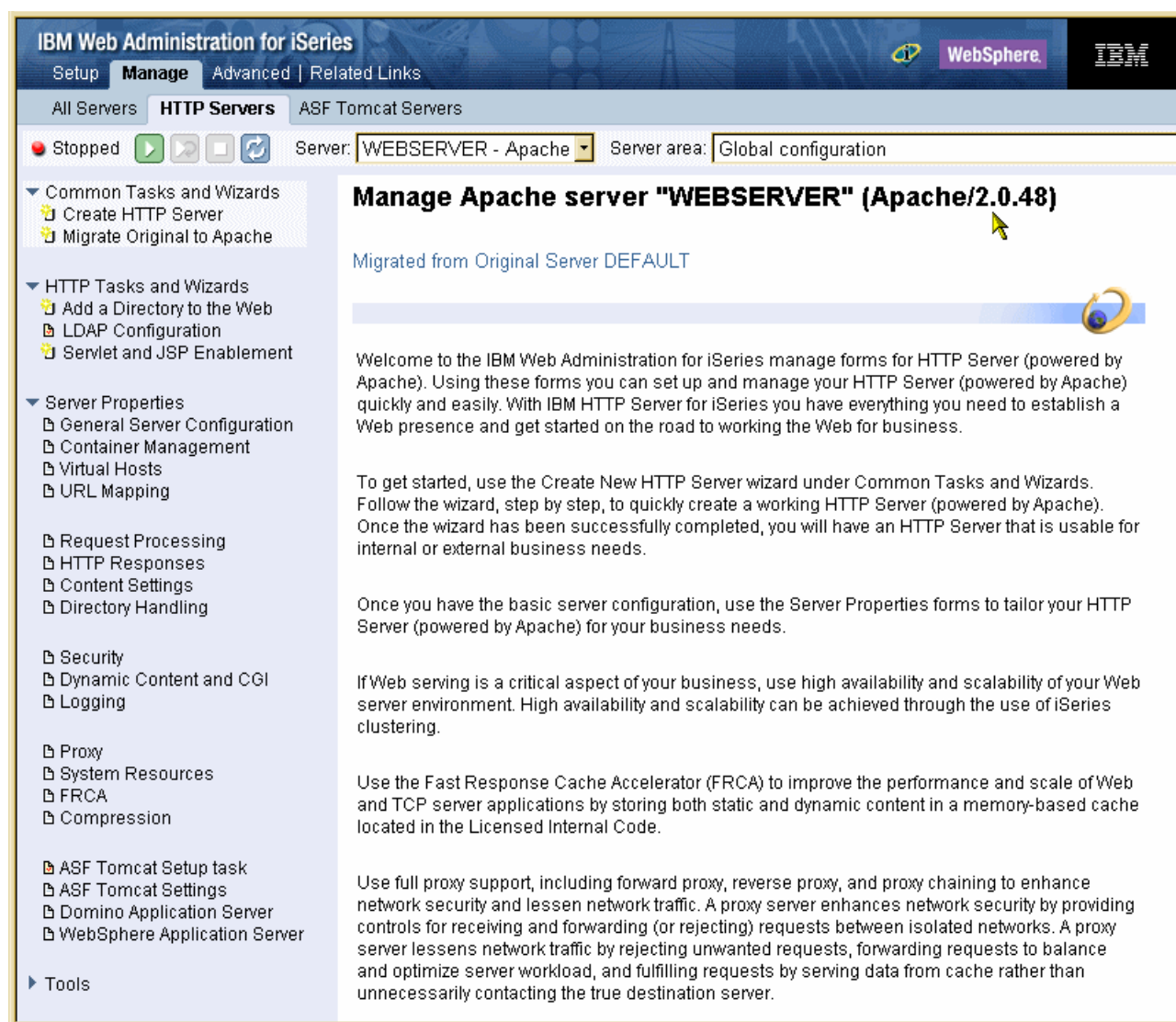


Figure 8-7 Manage Apache server WEBSERVER

8.2.3 Result: HTTP Server (powered by Apache) configuration

The migrated file included the HTTP Server (powered by Apache) directives shown in Example 8-3. To see the configuration on your own system, select any option under Server Properties in the Manage Apache server WEBSERVER as shown in Figure 8-7 and click the **Preview** button in the lower right corner of the panel.

Example 8-3 Directives in the configuration file

```
HTTP server:    WEBSERVER
Selected file:  /www/webserver/conf/httpd.conf

1  LogFormat "%h %l %u %t \"%r\" %>s %b" common
2  LiveLocalCache Off
3  <Location />
4      <LimitExcept GET HEAD OPTIONS POST TRACE>
5          Order Allow,Deny
6          Deny From all
7      </LimitExcept>
8  </Location>
9  Options -ExecCGI -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes
-MultiViews
10  DefaultType www/unknown
11  Listen *:80
12  ErrorLog Off
13  MaxKeepAliveRequests 5
14  Timeout 120
15  KeepAliveTimeout 4
+  AccessFileName .htaccess
16  DirectoryIndex welcome.html index.html
17  AddType text/plain .java
18  AddType image/bmp .bmp
19  AddType image/gif .gif
20  AddType text/html .htm
21  AddType text/html .html
22  CGIConvMode %%MIXED/MIXED%%
23  IndexOptions -DescriptionWidth -FancyIndexing -FoldersFirst -IconHeight
-IconsAreLinks -IconWidth -IgnoreCase -IgnoreClient -NameWidth -NameMinWidth
-ScanHTMLTitles -SelectiveDirAccess -ShowSmallFileBytes -ShowOwner -SuppressColumnSorting
-SuppressDescription -SuppressHTMLPreamble -SuppressIcon -SuppressLastModified
-SuppressRules -SuppressSize -TrackModified -VersionSort
24  HeaderName README
25  AddIconByType "/QIBM/HTTPSVR/Icons/text.gif" text/*
26  AddIconByType "/QIBM/HTTPSVR/Icons/sound.gif" audio/*
27  AddIconByType "/QIBM/HTTPSVR/Icons/movie.gif" video/*
28  AddIconByType "/QIBM/HTTPSVR/Icons/image.gif" image/*
29  AddIconByType "/QIBM/HTTPSVR/Icons/compress.gif" application/x-gzip
30  AddIconByType "/QIBM/HTTPSVR/Icons/compress.gif" application/x-compress
31  AddIconByType "/QIBM/HTTPSVR/Icons/binary.gif" application/*
32  AddIconByType "/QIBM/HTTPSVR/Icons/html.gif" text/html
33  AddAltByType "text" text/*
34  AddAltByType "au" audio/*
35  AddAltByType "vid" video/*
36  AddAltByType "img" image/*
37  AddAltByType "gzip" application/x-gzip
38  AddAltByType "Z" application/x-compress
39  AddAltByType "bin" application/*
40  AddAltByType "html" text/html
41  <Directory />
42      Order Allow,Deny
43      Deny From all
44  </Directory>
45  <Directory /QIBM/ProdData/OS400/SQL/Samples/>
46      Allow From all
47  </Directory>
48  <Directory /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/>
49      Allow From all
```

```
50      <FilesMatch ^Welcome\.html$>
51          Allow From all
52      </FilesMatch>
53  </Directory>
54  <Directory /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/Welcome.html/>
55      Allow From all
56  </Directory>
57  AliasMatch ^/$ /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/Welcome.html
58  AliasMatch ^/sample/(.*) /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/$1
59  AliasMatch ^/QIBM/ProdData/OS400/SQL/Samples/$ /QIBM/ProdData/OS400/SQL/Samples/
```

Now it's up to you to decide if you want to use this configuration. Be sure to test, test, and test again.

As shown in the previous example, the migrated HTTP Server (powered by Apache) configuration file contains some directives that you may not be able to decipher. The option always exists for you to create, from scratch, a new HTTP Server (powered by Apache) configuration file. You may find it easier to understand and, therefore, easier to extend in the future as your Web application grows.

8.3 Testing your migration

After a successful migration, you must test your new HTTP Server (powered by Apache) configuration. Before you start the new server, you need to change either the IP address or the port used by the HTTP server because the migration utility does not change this setting. Both servers are using the same IP address and port. The easy way is to change the port in the Apache server. Of course, you can always end the HTTP Server (original) instance for the duration of your test if this is feasible.

If you have any problems with the server starting or other failures, review Chapter 13, “Problem determination: When things do not go as planned” on page 323.

You can perform testing of your Web site and application at many different levels:

► Functionality

- Does the new server faithfully send all the static information such as Hypertext Markup Language (HTML), graphic images, and other media?
- Does the new server faithfully create dynamic data to be displayed within the HTML via Common Gateway Interface (CGI), Net.Data, and SSI?
- Did the migration of WebSphere Application Server directives allow Web application serving from your new server?

► Availability

Stress the new server environment to see if you can break it. Sometimes “timing windows” in Web applications that normally are not seen due to the performance characteristics of the HTTP Server (original) may be found when the same application is served from your HTTP Server (powered by Apache).

► Performance

- In general, you should expect your new HTTP Server (powered by Apache) to perform at about the same level as the HTTP Server (original) or better. Measure this. If a portion of your Web application is not performing at about the same level, determine why.

- It is possible that your migrated configuration file may not perform as well as though you had built the configuration from scratch. This is because the migration wizard may be forced to explicitly map original configuration directives to one or more Apache configuration directives in an attempt to faithfully reproduce function. This, in turn, may affect performance. In some cases, you may want to rewrite portions of migrated Apache configuration files after you have a better understanding of the ramifications of your actions.
- You may also want to take the opportunity to improve the performance of your new HTTP Server (powered by Apache) Web application. See Chapter 10, “Getting the best performance from HTTP Server (powered by Apache)” on page 223, for more information and guidance. Specifically two features that are only available with the HTTP Server (powered by Apache) stand out that could, depending on your Web application, improve your performance. One is Fast Response Cache Accelerator (FRCA) (see 10.6, “Fast Response Cache Accelerator” on page 281). The other is data compression (see 10.4, “Increasing throughput with compression” on page 240).

Tip: After the migration is completed, check the `KeepAliveTimeout` value. The corresponding directive for the HTTP Server (original) server is `PersistTimeout`. Under the HTTP Server (original), the default value for this directive was 4 seconds. After the migration, the `KeepAliveTimeout` is also set to 4 seconds. In many situations, this can cause problems. You should set this value to at least 60 seconds. We recommend that you set the value to 300 seconds.

► Security

- Test all forms of authentication and authorization. See 6.2, “Basic authentication” on page 103.
- Test your new servers’ handling of digital certificates – both client and server. See the sections starting with 6.4, “Encrypting your data with SSL and TLS” on page 127.
- Test all forms of proxy processing with your new server. See 6.5, “Proxy server: Protecting direct access” on page 142.
- Verify that access to i5/OS resources is performed under the expected user profile.

Note: The HTTP Server (original) supported two methods for defining protection setups. One was called a *named protection* and allowed an administrator to group a set of protection directives under a name. This name was then assigned to a protected resource. The second method was called an *inline protection setup*. This approach required all protection directives to be specified for each protected resource even if they were the same for all resources. The HTTP Server (powered by Apache) supports only the equivalent to the HTTP Server (original) inline protection setup. Therefore, the migration wizard converts named protection configurations to individual protection directives for each protected resource (context).



Web application serving

The Web is changing every aspect of our lives, but no area is undergoing as rapid and significant a change as the way businesses operate. As businesses incorporate Internet technology into their core business processes, they start to achieve real business value.

Today, large and small companies are using the Web to communicate with their partners, to connect to their back-end data systems, and to transact commerce. This is *On Demand Business*, where the strength and reliability of traditional IT meet the Internet.

But this On Demand Business world is supported by more than static Hypertext Markup Language (HTML) pages and images files. To use the Web for business, static HTML pages are not enough. Special applications are required to process a user's input and integrate the Web server with other information systems and data. The programs that extend the Web server beyond static content are called *Web applications*. Usually, Web applications use the data supplied by the HTML form, process the data, and then return the result as a Web page. The data processing can be achieved by a wide range of application environments and application programming languages.

Depending on the Web presence required for your environment, you can:

- ▶ Serve static pages for Web presence using HTTP Server (powered by Apache)
- ▶ Use the HTTP Server (powered by Apache) with Common Gateway Interface (CGI), Net.Data, or other third-party products for data access

For more information, see Chapter 7, "Serving dynamic data" on page 157.

- ▶ Have a Java application running with WebSphere Application Server or Apache Software Foundation (ASF) Jakarta Tomcat for a more powerful On Demand Business presence
- ▶ Install one of several available editions of the IBM WebSphere Application Server to support applications based on Java 2 Enterprise Edition (J2EE)

As shown in Figure 9-1, there are many options to create, serve, and manage one On Demand Business application using the iSeries server. The iSeries server aggressively supports the transformation of business applications to an On Demand Business model, while minimizing disruption within the enterprise environment. It has business-proven values (reliability, security, scalability, low cost of ownership) that support the latest enabling technologies for On Demand Business. In combination, these two qualities make the iSeries server an excellent choice for extending existing applications and deploying new solutions.

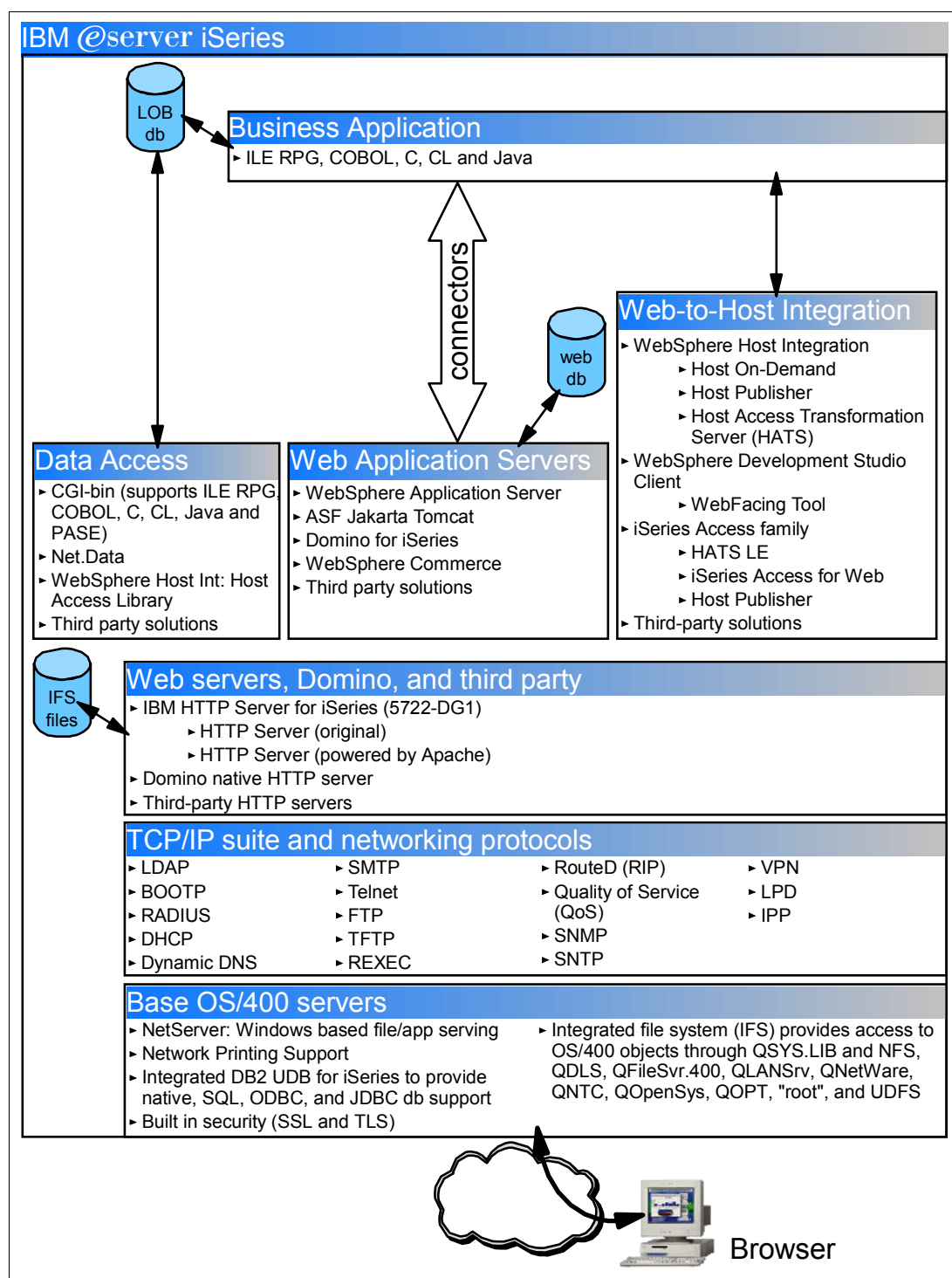


Figure 9-1 iSeries On Demand Business environments

9.1 Web application servers for the iSeries server

This section helps to identify the technical differences between WebSphere Application Server and Apache Software Foundation's Jakarta Tomcat Web application servers.

WebSphere Application Server Version 5.0 and 5.1

WebSphere Application Server Version 5.0 and 5.1 are available in three editions:

- ▶ **IBM WebSphere Application Server 5.0 and 5.1 for iSeries (Base Edition):** This edition ships as a Licensed Program Offering (LPO). It provides support for Java servlets, JavaServer Pages (JSP), Java Message Service (JMS), and Enterprise JavaBeans (EJBs). It also supports core Web Services standards such as Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Web Services Invocation Framework (WSIF) for developing dynamic solutions for On Demand Business.
- ▶ **IBM WebSphere Application Server Network Deployment 5.0 and 5.1 for iSeries (Network Deployment Edition):** This edition ships as an LPO. It delivers world-class caching, high availability, and industry-leading Web services support on top of the base WebSphere Application Server foundation.
- ▶ **IBM WebSphere Application Server 5.0 and 5.1 – Express for iSeries:** This edition offers a low-cost, easy to use, out-of-the box solution that supports simple, dynamic Web sites based on the Java Servlets, JSPs, and Web services technologies. In addition, it is now available at no additional charge with i5/OS V5R3M0 and later releases.

WebSphere Application Server Version 3.5 and 4.0

The WebSphere Application Server is a Java-based servlet engine that is built on top of the native Java Virtual Machine (JVM) on the iSeries server. It provides Java servlet application programming interface (API) support, which is defined by Sun Microsystems. WebSphere Application Server for iSeries Versions 3.5 and 4.0 have these editions:

- ▶ **Advanced Single Server Edition (Version 4.0 only):** This edition lets you use Java servlets, JSP, and XML to quickly transform static Web sites into vital sources of dynamic Web content. It also provides a high-performance EJB server to implement EJB components that incorporate business logic.
- ▶ **Advanced Edition (Versions 3.5 and 4.0):** This edition supports the same features as the Advanced Single Server. It also supports multiple machine topologies, distributed transactions, and transaction processing.
- ▶ **Standard Edition (Version 3.5 only):** This edition supports Java servlets, JSP, and XML.

End of Service: The end-of-service date for WebSphere Application Server Advanced Edition for iSeries V3.5.x was 31 December 2002. V3.5.x is not supported after this date. WebSphere Application Server Advanced Edition for iSeries V3.5.x is only supported on OS/400 V4R5 and V5R1. V3.5.x is not supported on OS/400 beyond V5R1.

WebSphere provides the application server, which includes:

- ▶ A servlet engine for running servlets and JSPs and, in the case of Advanced Edition, the container
The container is where Enterprise JavaBeans are deployed.
- ▶ An administrative server that is used to configure the servlets and JSPs in the servlet engine and the EJBs in the container
- ▶ An administrative console that is used to communicate with the administrative server

For additional information about WebSphere Application Server, see:

<http://www.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/>

Apache Software Foundation's Jakarta Tomcat

HTTP Server (powered by Apache) includes an industry-standard Java servlet and JSP engine based on technology from the ASF Jakarta Tomcat open source code base.

Lightweight and easy-to-use software extends the HTTP Server (powered by Apache). The iSeries server supports version 3.2.4 of ASF Jakarta Tomcat at V5R1, V5R2, and V5R3.

For additional information about ASF Jakarta Tomcat, see 9.2, "Apache Software Foundation's Jakarta Tomcat on iSeries" on page 197.

9.1.1 Comparing WebSphere Application Server and ASF Jakarta Tomcat

There are some technical differences between the application servers. Most of them relate to the components they support. Table 9-1 shows some of these differences.

Table 9-1 Web application servers: Versions and support

	IBM WebSphere Application Server				
	V 3.5 Standard and Advanced	V4.0 Single Server and Advanced	5.0 and 5.0 Express	5.1 and 5.1 Express	Tomcat 3.2.4
Servlets	2.2 + IBM extensions	2.2 + IBM extensions	2.3 + IBM extensions	2.3 + IBM extensions	2.2
JSP	1.1	1.1	1.1 & 1.2	1.2	1.1
JDK	1.2 and 1.3	1.3	1.3.1	1.4	1.2 and 1.3
EJB	1.0 (Advanced only)	1.1	1.1 and 2.0 (not for Express)	2.0 (not for Express)	Not supported
XML	Supported	Supported	Supported	Supported	Not provided directly as part of 5722-DG1, but is available as part of the IBM Toolbox for Java
Connection pooling	Supported	Supported	Supported	Supported	Not supported
Session support	IBM extension to Servlet 2.2	IBM extension to Servlet 2.2	IBM extension to Servlet 2.3	IBM extension to Servlet 2.3	Not supported. Sessions, as defined in servlet 2.2, are supported by ASF Jakarta Tomcat.
J2EE	No	Yes	Yes	Yes	No

Note: Although Tomcat 5 is not officially supported by IBM at V5R1, V5R2, or V5R3, we provide the steps that you can use in Appendix B, "Bringing Tomcat Version 5.5 to your iSeries server" on page 409.

9.1.2 When to use WebSphere Application Server versus ASF Jakarta Tomcat

Now that we identified some of the components supported by WebSphere Application Server and ASF Jakarta Tomcat, let's see how to choose one of the Web application servers to serve our solution for On Demand Business.

IBM's strategic Web application server is WebSphere Application Server. The latest version of WebSphere Application Server is Version 5.0. It includes three editions for iSeries customers:

- ▶ WebSphere Application Server V5.0 for iSeries (Base Edition)
- ▶ WebSphere Application Server Network Deployment V5.0 for iSeries (Network Deployment Edition)
- ▶ WebSphere Application Server Express for iSeries

These three editions of WebSphere Application Server V5 support servlets, JSP, EJB, and much more. Customers who require a robust and scalable Web application server select WebSphere Application Server. WebSphere Application Server is a chargeable product.

ASF Jakarta Tomcat on iSeries is the Web servlet and JSP container engine. This servlet engine is free of charge and is included with the IBM HTTP Server for iSeries. Table 2-2 on page 20 for details. The iSeries server supports ASF Jakarta Tomcat Version 3.2.4.

When to use WebSphere Application Server 4.0 Single Server Edition

Use the Single Server Edition when:

- ▶ You need to deploy solutions for On Demand Business that are J2EE compliant.
- ▶ Your application requires full Web services support.
- ▶ Your application benefits from EJB reloads.
- ▶ Your application requires Extensible Stylesheet Language (XSL) support.

When to use WebSphere Application Server 4.0 Advanced Edition

This is the same as 4.0 Standard Edition plus. Use it when:

- ▶ Your environment requires load balancing.
- ▶ You need real-time information about the On Demand Business application's behavior in terms of response time and access, because this edition includes the component Resource Analyzer.
- ▶ The application requires some degree of partitioning.

When to use WebSphere Application Server – Express for iSeries 5.0 or 5.1

Use WebSphere Application Server – Express for iSeries Version 5.1 when:

- ▶ You are looking to deploy your first Web-based application on iSeries or to deploy simple Web-based applications.
- ▶ You are looking to deploy Web-based applications developed with WebSphere Development Studio Client for iSeries.
- ▶ You have currently deployed applications in WebSphere Application Server Version 3.02 or Version 3.5, Standard Edition for AS/400, or the ASF Tomcat Web application server for iSeries.

When to use WebSphere Application Server V5.0 or 5.1 (Base Edition)

Use WebSphere Application Server V5 (Base Edition) when:

- ▶ You require full J2EE support.
- ▶ You need support for Java Servlets, JSPs, Java Message Service (JMS), and EJBs.
- ▶ Your environment uses core Web services standards such as XML, SOAP, WSDL, and WSIF for developing dynamic solutions for On Demand Business.

When to use WebSphere Application Server Network Deployment V5.0 or 5.1

Use this when you need all of the requirements as stated for the (Base Edition) and:

- ▶ You need caching support.
- ▶ You require high availability.
- ▶ You need industry leading Web services support on top of the base WebSphere Application Server foundation.

Note: All three versions of WebSphere Application Server Version 5.0 and 5.1 have new and added features that are too numerous to list. We recommend that you go to the following Web site and review the documentation for these products:

<http://www.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/>

After carefully reviewing this information, you can make your decision based on what you need for your business and environment. You should always try to deploy the latest WebSphere Application Server version to benefit from the latest enhancement.

When to use ASF Jakarta Tomcat

Some iSeries customers want a basic, *no-cost* Web application server that supports servlets and JSP. Relying on IBM HTTP Server (powered by Apache) as its Web server, ASF Jakarta Tomcat provides a basic Web application server for iSeries customers. Use ASF Jakarta Tomcat when:

- ▶ Your application is based on servlets, JSP, and XML files.
- ▶ Your application does not require EJB support.
- ▶ Your application does not require any database connection manager mechanism.
- ▶ Your application does not require any specific security mechanism, for example Secure Sockets Layer (SSL).
- ▶ Your solution for On Demand Business does not require a load balancing implementation.
- ▶ Your solution for On Demand Business does not require scalability.

ASF Jakarta Tomcat is the newest component in the solution for On Demand Business provided by the iSeries server. In the following section, you learn more about this new member.

For a detailed functional comparison of each of these WebSphere Application Server editions, see Chapter 5 in *WebSphere for the IBM @server iSeries Server Buying and Selling Guide*, REDP-3646.

For more information about WebSphere Application Server for iSeries, go to:

<http://www.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/>

For WebSphere Application Server for Linux on iSeries, see:

<http://www.ibm.com/servers/eserver/iseries/linux/websphere>

9.2 Apache Software Foundation's Jakarta Tomcat on iSeries

This is only supported by HTTP Server (powered by Apache).

ASF Jakarta Tomcat is a servlet engine container that supports servlets, JSP, and Web Application Archive (WAR) files. This servlet engine is developed and released under the Apache Software Foundation license. It is integrated in the iSeries server by the IBM HTTP Server for iSeries base code (see Table 2-2 on page 20 for the packaging details). ASF Jakarta Tomcat requires a Java Runtime Environment (JRE) that has conformity with JRE 1.1 or later, including any Java 2 platform system.

Note: If you want to go directly to an example of application serving using ASF Jakarta Tomcat, see 9.3.2, “In-process Tomcat configuration” on page 203.

The iSeries server supports Version 3.2.4 on V5R1, V5R2 and V5R3. We provide an example of how to set up Tomcat 5.5 on your V5R3 iSeries server in Appendix B, “Bringing Tomcat Version 5.5 to your iSeries server” on page 409.

For ASF Jakarta Tomcat to work with the HTTP Server (powered by Apache), it needs an *agent* that resides in the HTTP server and sends it a servlet request. This agent is the Web server plug-in, *jk_module*. It allows the communication between the HTTP Server (powered by Apache) and the ASF Tomcat servlet engine. It must be included in the HTTP configuration file with the LoadModule directive:

```
LoadModule jk_module /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM
```

Although the ASF Tomcat servlet engine is integrated into HTTP Server (powered by Apache), this does not mean that the servlet engine needs to run in the same process as the HTTP server. ASF Jakarta Tomcat can be configured to run:

- **In-process:** ASF Jakarta Tomcat and HTTP Server (powered by Apache) run in the same process and communicate through a Java Native Interface (JNI).
- **Out-of-process:** ASF Jakarta Tomcat and the HTTP Server (powered by Apache) run in separate process (even on separate systems) and communicate through Transmission Control Protocol/Internet Protocol (TCP/IP) sockets. The ASF Tomcat server process runs in the QSYSWRK subsystem.

Running in-process or out-of-process implies some differences as shown in Table 9-2.

Table 9-2 Differences between running Tomcat in-process and out-of-process

In-process	Out-of-process
Uses the jk_module module with the Java invocation API to communicate with the HTTP server.	Uses the jk_module to communicate with the HTTP server.
The HTTP server and ASF Tomcat servlet engine communicate through a JNI.	The HTTP server and ASF Jakarta Tomcat communicate through TCP/IP sockets.
Does not require a new protocol.	Requires a new protocol to communicate (ajp12 and ajp13).
ASF Tomcat server runs in the same JVM as the HTTP server.	ASF Tomcat server runs in its own JVM.
Uses the same security implementation configured by the HTTP server.	Uses its own container managed security implementation.
Works with the SSL configuration of the HTTP server.	The communication between the HTTP server and ASF Jakarta Tomcat does <i>not</i> support SSL.

Before you start either the in-process or out-of-process configurations, identify the information listed in Table 9-3 since it is required during the ASF Jakarta Tomcat configuration process. This table also helps to identify some of the differences between running Tomcat in-process and out-of-process.

Table 9-3 ASF Jakarta Tomcat required parameters for both in-process and out-of-process

Parameter	In-process	Out-of-process
ASF Tomcat server	Not required	Any name for the servlet engine
ASF Tomcat home directory	Usually the HTTP server root	Any directory; the default is /ASFTomcat/server_name
Servlet engine configuration file	/HTTP_home/conf/server.xml	/Tomcat_home/conf/server.xml
Java version (JDK)	1.2 or 1.3	1.2 or 1.3
URLs (Mount points)	Uniform Resource Locator (URL) paths for your application	URL paths for your application
Application contexts	Link between URL and your application directory; similar to pass or alias directive used by the HTTP server	Link between URL and your application directory; similar to pass or alias directive used by the HTTP server
IP address	Not required	The Internet Protocol (IP) address used by the servlet engine to communicate with the HTTP server
Port	Not required	The port used by the servlet engine to communicate with the HTTP server
Server type	Not required	The protocol used by the servlet engine to communicate with the HTTP server, AJP12 or AJP13
Server userid	Not required	The user ID used to start the servlet engine

9.2.1 ASF Jakarta Tomcat directory structure

This servlet engine runs under its own directory structure. This directory structure is used by the HTTP server since it is included into the HTTP configuration file. The directory structure can be located in the root or QOpenSys file systems. Table 9-4 shows the directory structure used by ASF Jakarta Tomcat.

Table 9-4 ASF Jakarta Tomcat directory structure

ASF Tomcat directory	Description
tomcat_home	The tomcat_home directory is the base directory for ASF Tomcat. The tomcat_home directory can be located in the root or QOpenSys file systems. For an in-process ASF Tomcat configuration, the default tomcat_home directory is set to the HTTP server directory (/www/server_name/). For an out-of-process ASF Tomcat configuration, the default tomcat-home directory is set to /ASFTomcat/tomcat_server_name/. Within the tomcat_home directory, there are subdirectories for logs and configuration information.
tomcat_home/webapps	This directory contains WAR files if you have them. All WAR files are expanded and subdirectories are added as contexts.

ASF Tomcat directory	Description
tomcat_home/webapps/ROOT	This directory is required by ASF Tomcat. This directory is required to support the servlet 1.1 specification.
tomcat_home/webapps/app1	This directory is known as a <i>document base directory</i> . You may have several document base directories under the webapps directory. These represent and map a directory structure to a servlet or JSP application. The subdirectory app1 is your application directory.
tomcat_home/webapps/app1/WEB-INF	This directory contains the web.xml file for the application. The web.xml file contains the URL patterns and attributes for your servlets.
tomcat_home/webapps/app1/WEB-INF/classes	This directory contains any Java class files and associated resources that are required for your application. This directory is searched prior to the tomcat_home/webapps/app1/WEB-INF/lib directory for any servlet .class file that is specified in the URL.
tomcat_home/webapps/app1/WEB-INF/lib	This directory contains any JAR files and associated resources that are required for your application.
tomcat_home/conf	This directory contains the server.xml and workers.properties configuration files.
tomcat_home/logs	This directory contains all log files.
tomcat_home/work	This directory is automatically generated by ASF Tomcat as a place to store intermediate files.
java/lib	This directory is created as a place to put .jar and .class files that you want to add to the class path.

9.2.2 ASF Jakarta Tomcat directives

The ASF Jakarta Tomcat directives are used by the HTTP server to redirect the request of servlets, JSP, and WAR files to the servlet engine. Before using any of these directives, the `jk_module` module must be loaded. The directives are:

- ▶ **JkAsfTomcat:** This directive allows ASF Jakarta Tomcat to be turned off without deleting particular ASF Jakarta Tomcat directives from the HTTP Server (powered by Apache) configuration file. When this directive is set to *off*, it appears to the user as though ASF Jakarta Tomcat was never enabled.
- ▶ **JkLogFile:** The `JkLogFile` directive is used to describe the full path name of the `jk_module` log file. The log file describes the flows of header and data between the HTTP Server (powered by Apache) and the ASF Tomcat servlet engine. It does not contain information relative to what happens after a request is forwarded to the servlet engine. The specified log file is never purged or wrapped. The file may need to be periodically purged by the administrator.
- ▶ **JkLogLevel:** The `JkLogLevel` directive is used to describe the detail of logging that should occur to the log file defined by `JkLogFile`. The possible values for this directive are:
 - debug
 - info
 - error
 - emerg
- ▶ **JkMount:** The `JkMount` directive specifies which Uniform Resource Identifier (URI) contexts are sent to a ASF Jakarta Tomcat worker.

- ▶ **JkMountCopy:** The JkMountCopy directive indicates whether the base server mount points should be copied to the virtual server. Any mount points defined outside <VirtualHost> </VirtualHost> are inherited by the virtual host.
- ▶ **JkWorkersFile:** The JkWorkersFile directive is used to define the name of a file that contains configuration information (that describes how jk_module attaches to the ASF Tomcat servlet engine). There is no default. This directive must be specified or ASF Jakarta Tomcat will not function. The typical file name is *workers.properties*.

These directives are added into the HTTP configuration file when the ASF Jakarta Tomcat configuration is created. The directives may look like the following example:

```
LoadModule jk_module /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM
...
JkWorkersFile /tomcat_home/conf/workers.properties
JkLogFile     /http_serverhome/logs/jk.log
JkLogLevel    error
JkMount       /orderentry/* remote
```

The workers.properties file

The *worker* is the ASF Jakarta Tomcat instance that runs to serve servlets and JSP requests coming from the Web server or, in our case, coming from the HTTP Server (powered by Apache). This worker can run in-process or out-of process. It is specified in the JkWorkersFile directive and tells the HTTP Server (powered by Apache) how the ASF Jakarta Tomcat instance runs. This file contains entries of the following form:

`worker.list=<a comma or space separated list of worker name>`

Consider the following examples:

```
worker.list=local, remote
worker.list=local remote
```

When starting, the Web server plug-in (jk_module) instantiates the workers whose names appear in the worker.list property. Each named worker should also have a few entries to provide additional information. Such things as the worker type, port, and other related information to the ASF Jakarta Tomcat process. The available workers types are:

- ▶ **ajp12:** This worker forwards requests to the out-of-process ASF Jakarta Tomcat process using the ajp12 protocol.
- ▶ **ajp13:** This worker forwards requests to the out-of-process ASF Jakarta Tomcat process using the ajp13 protocol.
- ▶ **jni:** This worker forwards requests to the in-process ASF Jakarta Tomcat process using Java Native Interface (JNI).

The differences between the ajp12 and ajp13 protocols are:

- ▶ The ajp13 protocol is a binary protocol and tries to compress some of the requested data.
- ▶ The ajp13 protocol reuses open sockets and leaves them open for future requests.
- ▶ The ajp13 protocol has special treatment for SSL information.

Defining workers of a certain type should be done with the following property format:

```
worker.worker name.type=worker type
```

Consider this example for the ASF Jakarta Tomcat in-process mode:

```
worker.local.type=jni
```

Here *local* is the name of this worker.

Each of these workers has its own group of properties that define the ASF Jakarta Tomcat attributes such as ports, host names, classpaths, and so on. The attributes are different if running in-process or out-of-process and if using ajp12 or ajp13 types. Depending on your ASF Jakarta Tomcat run mode, the workers.properties file should have entries like the ones shown here.

The following workers.properties file specifies the in-process mode, with Java 1.2 and Tomcat home directory /itso/itso07:

```
worker.list=local
worker.local.type=jni
worker.local.cmd_line=-config
worker.local.cmd_line=/itso/itso07/conf/server.xml
worker.local.sysprops=java.version=1.2
worker.local.sysprops=tomcat.home=/itso/itso07
worker.inprocess.stdout=/itso/itso07/logs/jvmstdout.txt
worker.inprocess.stderr=/itso/itso07/logs/jvmstderr.txt
worker.local.class_path=QIBM/ProdData/HTTP/java/lib/webserver.jar
```

The following workers.properties file specifies out-of-process using ajp13 protocol, on port 8009, and running in the local host in the iSeries server:

```
worker.list=remote
worker.remote.type=ajp13
worker.remote.port=8009
worker.remote.host=localhost
```

When you create the ASF Tomcat servlet engine, using the ASF Jakarta Tomcat wizard provided with the HTTP Server (powered by Apache) server, those entries are created in the workers.properties file for you. At this point, this information is only supplied as a reference.

If you want to learn more about the workers.properties file and its properties, refer to the Tomcat Workers How To on the Apache Software Foundation's Web site at:

<http://jakarta.apache.org/tomcat/tomcat-3.2-doc/Tomcat-Workers-HowTo.html>

9.2.3 ASF Jakarta Tomcat authorities

To run ASF Jakarta Tomcat, there are some authority requirements that the user running the server must accomplish. The security considerations are related to the way the ASF Tomcat servlet engine runs, either in-process or out-of-process.

This information is kept current in the Documentation Center's document "User profiles and required authorities for the HTTP Server (powered by Apache)". To find this document, see the following Web site:

<http://www-1.ibm.com/servers/eserver/iseries/software/http/docs/doc.htm>

When you reach this site, select the document for the version you are using, either V5R1 or V5R2. Inside the IBM HTTP Server for iSeries Documentation Center, click **HTTP Server (powered by Apache)** → **Reference** → **User profiles and required authorities**.

9.2.4 ASF Jakarta Tomcat log files

ASF Jakarta Tomcat has its own set of logs files used to track day-by-day operations and error messages for problem determination. Each time the ASF Tomcat servlet engine is started, a set of log files is generated. They are all specified on the configuration file `server.xml`. The files are located under the directory structure `/ASFTomcat/server_name/logs`, used by ASF Jakarta Tomcat. Under this directory structure, you see the files shown in Table 9-5.

Table 9-5 ASF Jakarta Tomcat log files

Log file	Description
<code>jasper.log</code>	This log file contains messages resulting from trying to start or run JSPs.
<code>servlet.log</code>	This log file contains messages generated as a result of a servlet running in the ASF Tomcat servlet engine. When a servlet is initialized, a <code>ServletConfig</code> object is provided to the servlet. Contained within the <code>ServletConfig</code> object is a <code>ServletContext</code> object that provides methods for a servlet to communicate to the Servlet container, which is Tomcat in this case. In the <code>ServletContext</code> object is a log method that allows Web applications to log to the <code>servlet.log</code> file.
<code>tomcat.log</code>	This log file contains ASF Tomcat servlet engine messages.
<code>jvmstderr.txt</code>	This log file can contain messages from any Java code that does a <code>System.err.println()</code> .
<code>jvmstdout.txt</code>	This log file can contain messages from any Java code that does a <code>System.out.println()</code> .
<code>jk.log</code>	This file contains messages generated by <code>jk_module</code> .

It is important to note that the `jk.log` file is not erased or regenerated when the ASF Tomcat engine starts. Messages are appended to this file and the size of this file can grow quite large if errors are logged. You should periodically monitor the size of this file and reduce its size. By default, the `jk.log` is set to the logs directory under the HTTP `server_home` (`/www/server_name/logs/`).

9.3 In-process implementation with ASF Jakarta Tomcat

This section serves a simple servlet using an in-process ASF Jakarta Tomcat configuration.

9.3.1 Creating HTTP Server (powered by Apache)

Use the Create New HTTP Server wizard to create a new HTTP Server (powered by Apache) that will be used for your in-process Tomcat configuration. During the course of the wizard, it asks you for information. Use the values specified in Table 9-6.

Table 9-6 In-process Tomcat: Create New HTTP Server wizard required parameters

HTTP Server wizard parameter	Value
Server name	PBATICIN01
Server root	/tcp52d01/asfTomcat
Document root	/tcp52d01/asfTomcat/htdocs
On which IP address and TCP/IP port do you want your server to listen?	IP address: all Port: 8301
Do you want your new server to use an access log?	Yes

With the information you provide, the Create New HTTP Server wizard creates the basic HTTP configuration file to serve static pages from your document root. The confirmation page for values specified in Table 9-6 should appear similar to the example in Figure 9-2.



Figure 9-2 In-process Tomcat: Create New HTTP Server wizard confirmation page

Your HTTP Server (powered by Apache) is now ready to be tailored for ASF Jakarta Tomcat.

9.3.2 In-process Tomcat configuration

The ASF Tomcat servlet engine can be configured to run in-process:

1. As shown in Figure 9-3 in the Server list, make sure your server name is selected. From the Server area list, select **Global configuration**.
2. In the left pane, select **Servlet and JSP Enablement**.
3. In the first Servlet and JSP Enablement panel (Figure 9-3), click **Next** to start the wizard.

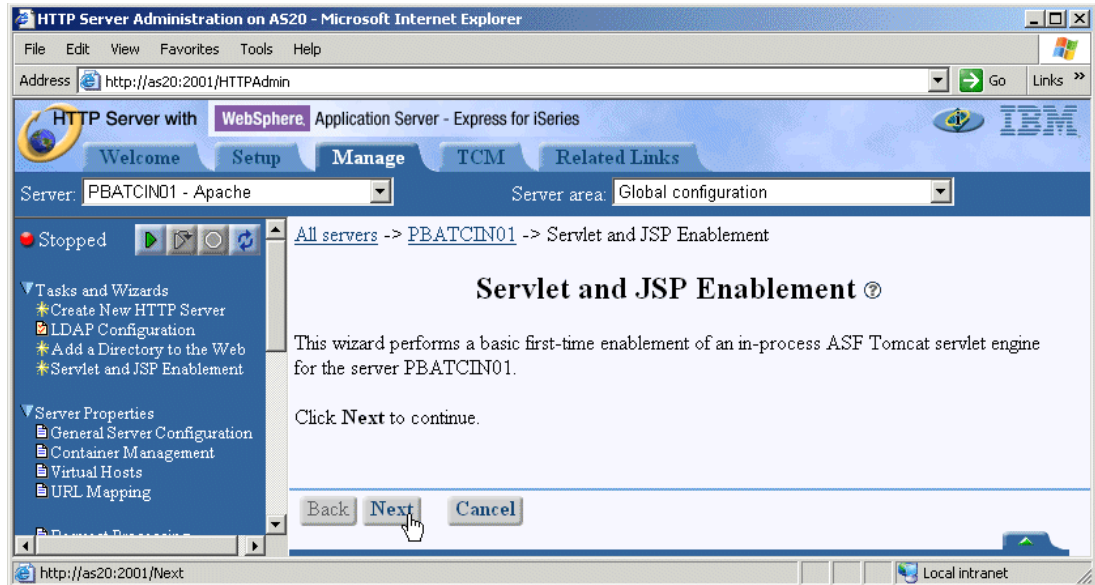


Figure 9-3 In-process Tomcat: Servlet and JSP Enablement wizard

4. As shown in Figure 9-4, select **I want to use a servlet or Java Server Page (JSP), and I either already have them or will provide them later**. Click **Next**.

Tip: An easy way to start the ASF Jakarta Tomcat configuration, directory structure, and activation process is to select **I want a sample ASF Tomcat in-process servlet engine configured for me**. This option creates an in-process configuration for two sample applications: a sample Calculator servlet and a sample Snoop JSP.

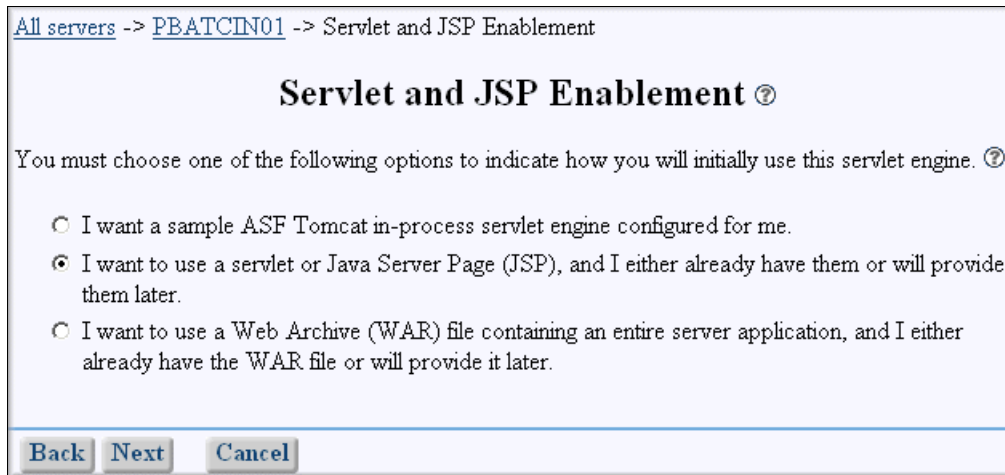


Figure 9-4 In-process Tomcat: Servlet and JSP Enablement wizard using your own servlet

5. As shown in Figure 9-5, complete these tasks:
 - a. Select **I want to use a servlet. I either already have a class or jar file containing the servlet or will provide it later.**
 - b. The Server class name field appears. Type the name of the sample servlet class. Replace the default MyServlet with CalculatorExample.
 - c. Click **Next**.

All servers -> PBATCIN01 -> Servlet and JSP Enablement

Servlet and JSP Enablement ?

Select the type of servlet to configure. ?

☒ I want to use a servlet. I either already have a class or jar file containing the servlet or will provide it later.

Servlet class name: ?

☐ I want to use a Java Server Page (JSP). I already have a JSP file or will provide it later

Figure 9-5 In-process Tomcat: Servlet and JSP Enablement wizard naming your servlet

6. As shown in Figure 9-6, click **Finish**. This page (and the next one) gives you good information about the URL to access your servlet and where in the integrated file system (IFS) to place it. You may want to take note of this information since we will use it in the next step.

All servers -> PBATCIN01 -> Servlet and JSP Enablement

Servlet and JSP Enablement ?

You have selected to provide your own servlet named **CalculatorExample**, it can be accessed using the URL **http://[HostName:Port]/app1/calculatorexample**.

When you click **Finish** your files will need to be placed in the following directories before starting the server:

- Your servlet class files will need to be placed in **/tcp52d01/asfTomcat/webapps/app1/WEB-INF/classes**. Your servlet jar files will need to be placed in **/tcp52d01/asfTomcat/webapps/app1/WEB-INF/lib**.

Figure 9-6 In-process Tomcat: Servlet and JSP Enablement wizard clicking Finish

7. Click **OK** on the confirmation page that follows.
8. To make your servlet name simpler than CalculatorExample, you can change the name used to invoke it. In this case, you can either use the *ASF Tomcat Setup Task* or *ASF Tomcat Settings* found in the left pane of the configuration options.
 - **ASF Tomcat Setup Task:** This multi-form task guides you through the various Apache configuration settings needed for using the ASF Tomcat servlet engine within a HTTP

server (powered by Apache). This is almost like a wizard that takes you through the process of changing the configuration options for your ASF Jakarta Tomcat.

- **ASF Tomcat Settings:** This single form allows you a single place in which to change the configuration options for your ASF Jakarta Tomcat.

We use the ASF Tomcat Settings because they provide an all-in-one form for configuration.

- As shown in Figure 9-7, in the left pane, select **ASF Tomcat Settings**.
- In the right panel, scroll down until you find the Application contexts table. Click **Configure**.

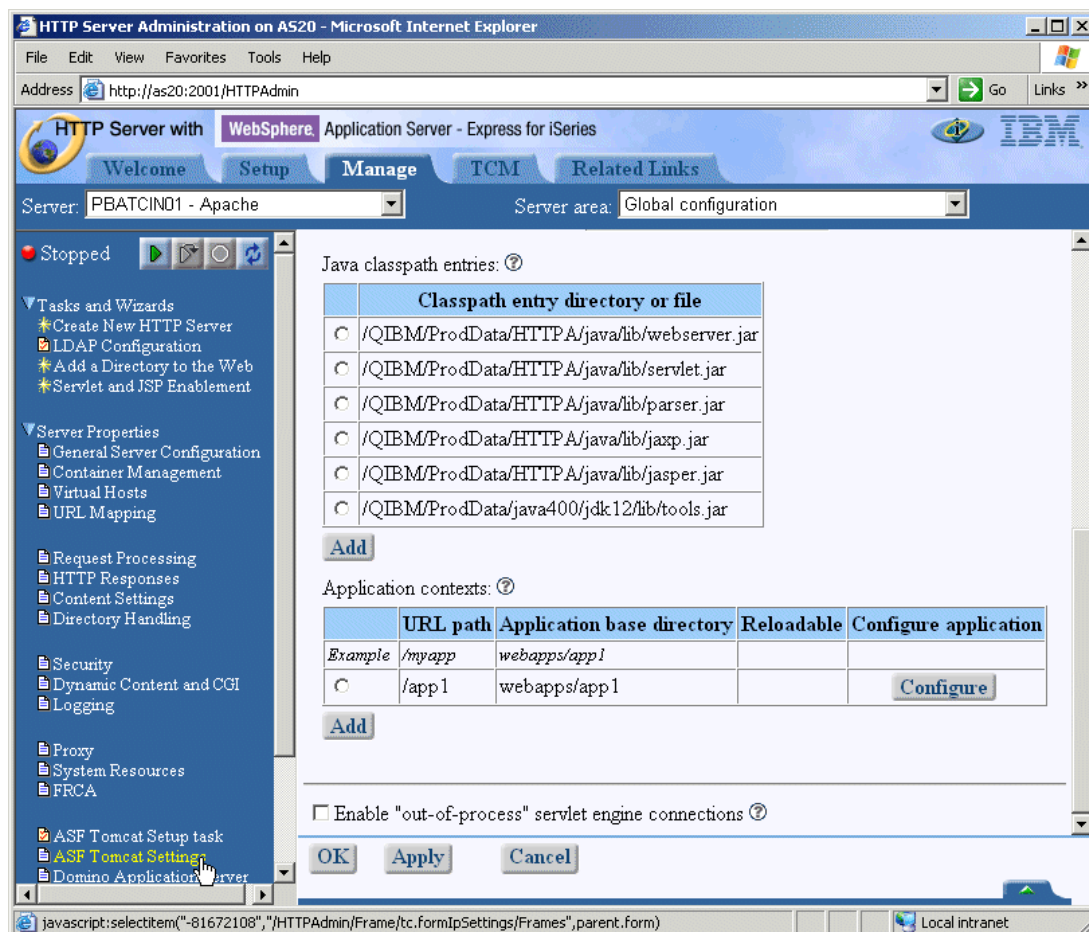


Figure 9-7 In-process Tomcat: ASF Tomcat settings

- c. In the ASF Tomcat Application Configuration form (Figure 9-8), follow these steps:
 - i. Select the row that contains the Servlet classname of CalculatorExample.
 - ii. Replace the URL pattern of /calculatorexample with the shorter /calc.
 - iii. Click **Continue**.
 - iv. Click **OK**.

ASF Tomcat Application Configuration for app1 - Microsoft Internet Explorer

File Edit View Favorites Tools Help

ASF Tomcat Application Configuration ?

Application base directory: /tcp52d01/asfTomcat/webapps/app1
 Application configuration file: /tcp52d01/asfTomcat/webapps/app1/WEB-INF/web.xml

Session object timeout: 30 Minutes ?

Servlet definitions: ?

	Servlet classname	URL patterns	Startup load sequence
Example	MainServlet	/Welcome	0 (Do not load)
Example	com.acme.otherapp	/*	1
<input checked="" type="radio"/>	CalculatorExample	/calc	0

Add Remove Move up Move down Continue

OK Apply Cancel

http://as20:2001/Continue Local intranet

Figure 9-8 In-process Tomcat: ASF Tomcat Settings updating URL patterns

- d. In the ASF Tomcat Settings form, click **OK**.
9. You have finished setting ASF Tomcat in the configuration file. Now continue with these steps:
 - a. Locate the **CalculatorExample.class** file in the /QIBM/ProdData/HTTPPA/admin directory.
 - b. Copy this file to the /tcp52d01/asfTomcat/webapps/app1/WEB-INF/classes directory.

10. Start your HTTP Server (powered by Apache) and run the servlet:

- a. Start your server PBATCIN01.
- b. Enter the following URL in your browser:

`http://as20:8301/app1/calc`

This opens the CalculatorExample servlet application as shown in Figure 9-9.

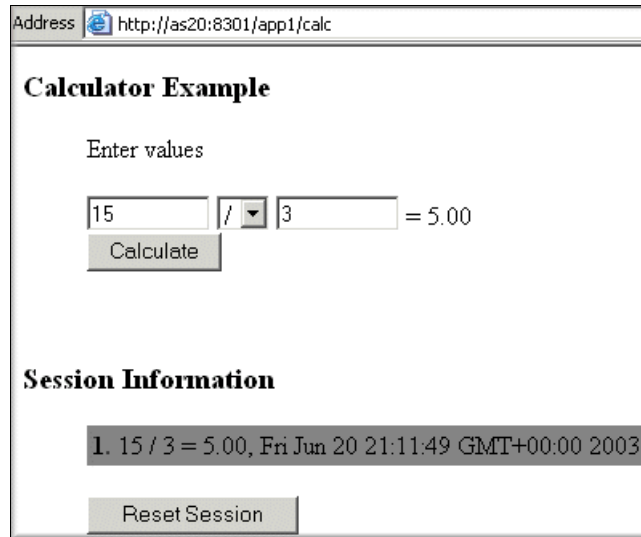


Figure 9-9 In-process Tomcat: CalculatorExample.class file running

9.4 Out-of-process implementation with ASF Jakarta Tomcat

This section shows how to serve a simple servlet using an out-of-process ASF Jakarta Tomcat configuration. The steps to configure ASF Jakarta Tomcat for out-of-process are:

1. Create the ASF Tomcat server.
2. Create the link between the HTTP and ASF Tomcat servers.
3. Test the out-of-process ASF Tomcat server.

This section expects that you have created a fairly standard HTTP Server (powered by Apache) with the characteristics specified in Table 9-7 or that you will use one of your own.

Table 9-7 In-process Tomcat: Create New HTTP Server wizard required parameters

HTTP Server wizard parameter	Value
Server name	PBATCOUT01
Server root	/tcp52d01/asfTomcatOut
Document root	/tcp52d01/asfTomcatOut/htdocs
On which IP address and TCP/IP port do you want your server to listen?	IP address: all Port: 8401
Do you want your new server to use an access log?	Yes

We serve the CalculatorExample.class servlet using this ASF Tomcat server. Before you start the configuration process, identify some data as shown in Table 9-8. This information is used to configure the ASF Tomcat server. It is also used to modify your HTTP Server (powered by Apache) to redirect certain URL patterns to the out-of-process ASF Tomcat server.

Table 9-8 ASF Jakarta Tomcat out-of-process parameters

Parameter	Value
ASF Tomcat server name	tomcat01
ASF Tomcat home directory	/tcp52d01/asfTomcatOut/TomcatHome
Server userid	QTMHHTTP
Java version (JDK)	1.2
IP address:port and protocol	*:8501 ajp13
URL path	/myapp
Application base directory	webapps/myapp
URL mount point	/myapp/*
Servlet classname	CalculatorExample
URL patterns	/calc

9.4.1 Creating the ASF Tomcat server

Using the out-of-process approach, you must create the ASF Tomcat server. This server is the one that receives the servlet or JSP request, processes it, and sends the response back to the HTTP server. The communication between the HTTP and ASF Tomcat server is through TCP/IP sockets. For this communication, you need to identify the IP address, port, and protocol both servers will use to communicate. This is the IP address and port the ASF Tomcat server works on. It is not related to the IP address and port used by the HTTP server to receive the user's request.

The following steps explain how to create an out-of-process ASF Tomcat server. Use the values specified in Table 9-8 for this configuration.

1. Start the administrative GUI and click the **Setup** tab as shown in Figure 9-10.
2. In the left pane, under Tasks and Wizards, select **Create ASF Tomcat Server**.

3. In the Out-of-Process Engine Creation panel (Figure 9-10), in the ASF Tomcat server name field, type your ASF Tomcat server name. In this example, we enter tomcat01. This is the name of the ASF Tomcat server process. This is also the name of the job that will start in the QSYSWRK subsystem when the ASF Tomcat server is started.

Click **Next**.

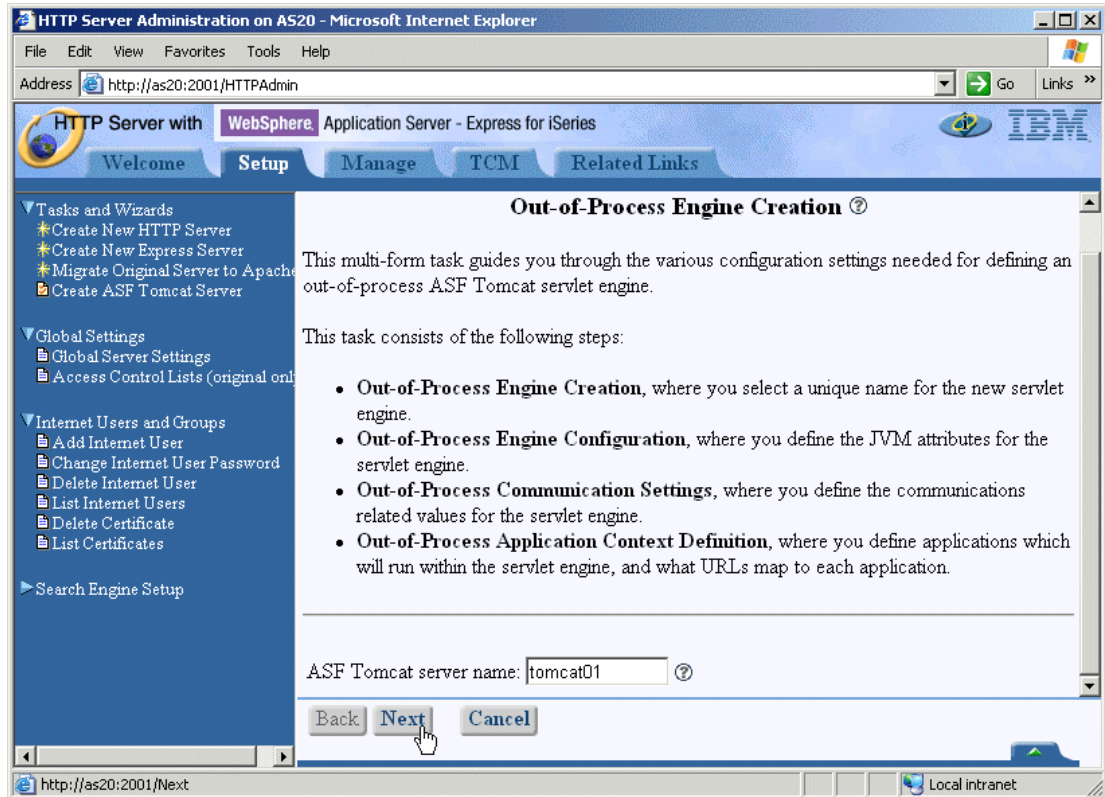


Figure 9-10 Out-of-process Tomcat: Creating an ASF Tomcat Server

4. The Out-of-Process Engine Configuration page opens as shown in Figure 9-11. Configure the user profile used to start the ASF Jakarta Tomcat process, the Java version used to start the server, and the home directory where ASF Jakarta Tomcat looks for property files and Web application files.
 - a. Enter the user profile you will use to start the ASF Tomcat server. For this example, we keep the default Server userid of QTMHHTTP.
 - b. Select the Java version this server will work with. In this example, we use Java version (JDK) **1.2**.
 - c. Enter the ASF Jakarta Tomcat home for your environment. For this example, we change the ASF Jakarta Tomcat home to be /tcp52d01/asfTomcatOut/TomcatHome.
 - d. Leave the defaults for the Java classpath entries.
 - e. Click **Next**.

Create ASF Tomcat Server

Out-of-Process Engine Configuration ?

ASF Tomcat server: TOMCAT01

You must define the various attributes of the Java Virtual Machine (JVM) which will be started within this servlet engine.

Server userid: ?

Java version (JDK): ?

ASF Tomcat home: ?

Java classpath entries: ?

	Classpath entry directory or file
<input type="radio"/>	/QIBM/ProdData/HTTP/java/lib/webserver.jar
<input type="radio"/>	/QIBM/ProdData/HTTP/java/lib/servlet.jar
<input type="radio"/>	/QIBM/ProdData/HTTP/java/lib/parser.jar
<input type="radio"/>	/QIBM/ProdData/HTTP/java/lib/jaxp.jar
<input type="radio"/>	/QIBM/ProdData/HTTP/java/lib/jasper.jar
<input type="radio"/>	/QIBM/ProdData/java400/jdk12/lib/tools.jar

Figure 9-11 Out-of-process Tomcat: Engine configuration

5. The Out-of-Process Communication Settings page (Figure 9-12) opens. Follow these steps:
 - a. Type the IP address the ASF Tomcat server will listen on. We enter All addresses.
 - b. Type the port number on which the ASF Tomcat server will listen. We type port 8501.

Note: The default value is normally 8009. Since this is a test system, we change the port to something unique for this shared iSeries server.

- c. Select the server type that the ASF Jakarta Tomcat will use to communicate with the HTTP server. We select **Binary (AJP13)**.
 - d. Click **Next** (not shown).

Create ASF Tomcat Server

Out-of-Process Communication Settings ?

ASF Tomcat server:	TOMCAT01
ASF Tomcat home directory:	/tcp52d01/asfTomcatOut/TomcatHome
Servlet engine configuration file:	/tcp52d01/asfTomcatOut/TomcatHome/conf/server.xml ?

You must define the IP address, TCP port and server type of this out-of-process ASF Tomcat servlet engine. The address and port defines how the server will listen for requests, and the server type defines the communications protocol it will use. These settings must match the ones defined for an out-of-process worker, within the configuration of the HTTP server wishing to use this servlet engine.

IP address: ?

Port: ?

Server type: ?

Figure 9-12 Out-of-process Tomcat: Communication settings

6. The Out-of-Process Application Context Definition panel (Figure 9-13) opens. Follow these steps:
 - a. Click **Add** to add a new row to the Application contexts table.
 - b. Enter your URL path. For this example, we use /myapp.
 - c. The Application base directory specifies the directory where the servlets and JSP are located. For this example, we use webapps/myapp. Under this directory, the ASF Jakarta Tomcat out-of-process wizard creates some files and directories required by ASF Jakarta Tomcat.
 - d. Click **Continue**.

Create ASF Tomcat Server

Out-of-Process Application Context Definition ?

ASF Tomcat server:

TOMCAT01

ASF Tomcat home directory:

/tcp52d01/asfTomcatOut/TomcatHome

Servlet engine configuration file:

/tcp52d01/asfTomcatOut/TomcatHome/conf/server.xml ?

You must identify the applications which will run within this out-of-process ASF Tomcat servlet engine. These applications are identified by the root URL path to be routed to them, and the application base directory where they reside (relative to the ASF Tomcat home directory.) Use the "Configure" button within the table below to further configure individual applications.

Application contexts: ?

	URL path	Application base directory	Reloadable	Configure application
Example	/myapp	webapps/app1		
+	/myapp	webapps/myapp	<input type="checkbox"/>	

Add

Remove

Move up

Move down

Continue

Figure 9-13 Out-of-process Tomcat: URL path and application base directory configuration

Note: If this is your first ASF Tomcat server, you see a message indicating that the web.xml file does not exist. This is normal. Continue with the next step.

7. The page shown in Figure 9-14 opens. Click **Configure**.

Application contexts: ?

	URL path	Application base directory	Reloadable	Configure application
Example	/myapp	webapps/app1		
+	/myapp	webapps/myapp		<div>Configure</div>

Note: Cannot find a WEB-INF/web.xml configuration file under the application base directory /tcp52d01/asfTomcatOut/TomcatHome/webapps/myapp. Use the "Configure" button to create one, unless you will be installing a .jsp or .war file.

Add

Figure 9-14 Out-of-process Tomcat: Missing web.xml configuration file

8. The ASF Tomcat Application Configuration page (Figure 9-15) opens. Follow these steps:
 - a. Click **Add** to add a new row to the Servlet definitions table.
 - b. Under Servlet classname, enter your Servlet classname. For this example, we use /CalculatorExample.
 - c. Under URL patterns, enter your URL path. For this example, we enter /calc.
 - d. Click **Continue**. You can add any number of servlets here.
 - e. Click **OK** (not shown). This closes the window and returns you to the Out-of-Process Application Context Definition window.
 - f. Click **Next** (not shown) to continue with the wizard.

ASF Tomcat Application Configuration ?

Application base directory: /tcp52d01/asfTomcatOut/TomcatHome/webapps/myapp

Application configuration file: /tcp52d01/asfTomcatOut/TomcatHome/webapps/myapp/WEB-INF/web.xml

Session object timeout: Minutes ?

Servlet definitions: ?

	Servlet classname	URL patterns	Startup load sequence
<i>Example</i>	MainServlet	/Welcome	0 (Do not load)
<i>Example</i>	com.acme.otherapp	/*	1
+	CalculatorExample	/calc	<input type="text" value="0"/>

Add Remove Move up Move down Continue

Figure 9-15 Out-of-process Tomcat: ASF Tomcat Application Configuration

9. The Out-of-Process Summary page (Figure 9-16) opens. Complete these steps:
 - a. Click **Finish** to create the out-of-process Tomcat server.
 - b. You receive a message that indicates that your ASF Tomcat servlet engine TOMCAT01 has been successfully created. Click **OK** to continue.

Create ASF Tomcat Server

Out-of-Process Summary ?

When you click **Finish** all necessary configuration files will be updated with these settings.

Java classpath entries:

ASF Tomcat server name: TOMCAT01

Server userid: QTMHHTTP

Java version (JDK): 1.2

ASF Tomcat home: /tcp52d01/asfTomcatOut/TomcatHome

Servlet engine configuration file: /tcp52d01/asfTomcatOut/TomcatHome/conf/server.xml

IP address: All addresses

Port: 8501

Server type: Binary (AJP13)

Classpath entry directory or file
/QIBM/ProdData/HTTP/java/lib/webserver.jar
/QIBM/ProdData/HTTP/java/lib/servlet.jar
/QIBM/ProdData/HTTP/java/lib/parser.jar
/QIBM/ProdData/HTTP/java/lib/jaxp.jar
/QIBM/ProdData/HTTP/java/lib/jasper.jar
/QIBM/ProdData/java400/jdk12/lib/tools.jar

Application contexts:

URL path	Application base directory	Reloadable	Configure application
/myapp	webapps/myapp	<input type="checkbox"/>	<input type="button" value="Configure"/>

Figure 9-16 Out-of-process Tomcat: Summary

Here is what the wizard has done for you. Under your ASF Tomcat home directory /tcp52d01/asfTomcatOut/TomcatHome, some directories and files were created as shown in Figure 9-17. The following actions were performed:

- a. Created the ASF Tomcat home directory /TomcatHome
 - b. Created the /conf configuration directory
 - c. Created the web.xml file
 - d. Created the /logs directory
 - e. Created the /webapps directory
 - f. Created the /myapp directory
 - g. Created the /WEB-INF directory
 - h. Created the /classes directory where the CalculatorExample.class file will be placed
10. Place your CalculatorExample.class file into the /TCP52D01/asfTomcatOut/TomcatHome/webapps/myapp/WEB-INF/classes directory.

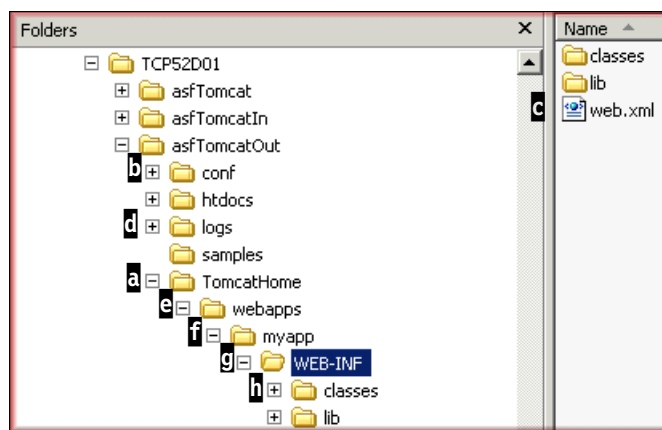


Figure 9-17 Out-of-process Tomcat: Directory structure

9.4.2 Creating the link between the HTTP and ASF Tomcat servers

The ASF Jakarta Tomcat and the HTTP Server (powered by Apache) configurations are now both created. Now you must include the ASF Tomcat server directives that will cause the servlet and JSP request made to the HTTP Server (powered by Apache) to be redirected to the ASF Tomcat engine.

1. Start the administrative GUI and click the **Manage** tab as shown in Figure 9-18.
2. From the Server list, select the HTTP server you want to work with. For this example, we select **PBATCOUT01**.
3. In the left pane, under Server Properties, select **ASF Tomcat Setup task**.

4. In the Apache Enablement panel on the right (Figure 9-18), complete these tasks:
 - a. Select **Enable servlets for this HTTP Server**.
 - b. In the Workers definition file field, enter the directory path where the workers file should be located. In this example, we use the directory suggested by the wizard, which is the configuration directory of the HTTP server.
 - c. Click **Next**.

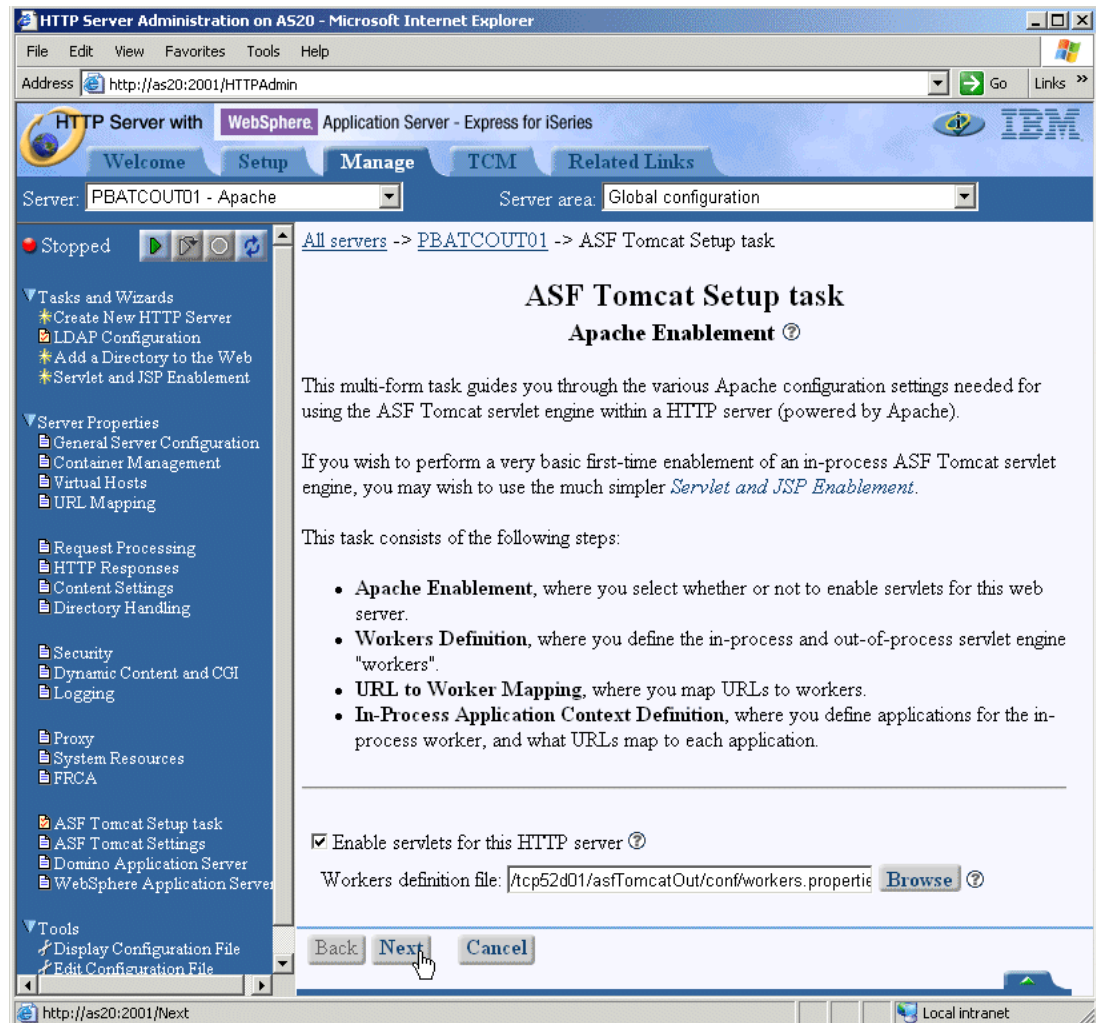


Figure 9-18 Out-of-process Tomcat: Apache enablement

5. The Workers Definition page (Figure 9-19) opens. Follow these steps:
 - a. Deselect the **Enable an “in-process” servlet engine** check box.
 - b. Select the **Enable “out-of-process” servlet engine connections** check box.
 - c. Click **Add** to add a row to the Out-of-process workers table.
 - d. In the Worker name field, type a name. For this example, we enter remote.
 - e. In the Worker type field, select the communications protocol version that will be used to communicate between the HTTP Server (powered by Apache) and the Tomcat server. For this example, we select **Binary (AJP13)**. The server type should be the same as you selected for the ASF Tomcat server engine creation. See Figure 9-12 on page 212.
 - f. In the Hostname:Port field, enter the IP address and port the on which Tomcat server listens. For this example, we enter localhost for the IP address and 8501 for the port. This value must be the same as the one that was used in Figure 9-12 on page 212.
 - g. Click **Continue**.
 - h. Click **Next** (not shown).

ASF Tomcat Setup task

Workers Definition ?

Workers definition file: /tcp52d01/asfTomcatOut/conf/workers.properties

☐ Enable an "in-process" servlet engine ?

☒ Enable "out-of-process" servlet engine connections ?

Out-of-process workers: ?

	Worker name	Worker type	Hostname:Port
<i>Example</i>	worker1	Binary (AJP13)	localhost:8009
<input checked="" type="radio"/>	remote	Binary (AJP13) ▼	localhost:8501 or...

Add
Remove
Move up
Move down
Continue

Figure 9-19 Out-of-process Tomcat: Configuring the Workers Definition file

6. The URL to Worker Mapping page (Figure 9-20) opens. Here you create the link between the URL path and the ASF Tomcat server servlet engine.
 - a. Click **Add** to add a new row in the URL mappings table.
 - b. In the URL (Mount point) field, enter the URL path used by the HTTP server to identify a servlet or JSP request and route the request to the ASF Tomcat server. For this example, we use `/myapp/*`. Therefore, each time the HTTP server receives a request from `http://hostname:port/myapp/*`, it sends the request to the ASF Tomcat server.
 - c. In the ASF Tomcat worker field, select the out-of-process worker that this HTTP server will work with. In this case, we select **remote (localhost:8501)**.
 - d. Click **Continue**.
 - e. Click **Next** (not shown).

ASF Tomcat Setup task

URL to Worker Mapping ?

You must define which URLs within the Apache configuration file are to be mapped to ASF Tomcat, and which predefined worker will process the request.

URL mappings: ?

	URL (Mount point)	ASF Tomcat worker
<i>Example</i>	<code>/myapp</code>	<code>inprocess</code>
<i>Example</i>	<code>/myapp/*</code>	<code>inprocess</code>
+	<input type="text" value="/myapp/*"/>	<input type="text" value="remote (localhost:8501)"/>

Add
Remove
Move up
Move down
Continue

Figure 9-20 Out-of-process Tomcat: URL to worker mapping

7. The Configuration Summary page appears. Review the information and click **Finish**.
8. Click **OK** to continue.

The ASF Tomcat Setup task wizard creates the workers.properties file under the HTTP home directory as shown in Figure 9-21.

```

Browse : /TCP52D01/asfTomcatOut/conf/workers.properties
Record :      1    of      10 by 18                      Column :      1      76 by 131
Control :

.....1.....2.....3.....4.....5.....6.....7.....8.....
*****Beginning of data*****
#
# ASF Tomcat workers definition file for IBM HTTP server (powered by Apache)
# Thu Jul 10 19:33:59 UTC 2003
#

worker.list=remote

worker.remote.type=ajp13
worker.remote.host=localhost
worker.remote.port=8501
*****End of Data*****

```

Figure 9-21 Out-of-process Tomcat: Workers.properties file

9.4.3 Testing the out-of-process ASF Tomcat server

Your configuration is now complete. Both the ASF Tomcat server and the HTTP server are configured to serve the On Demand Business application. Before you start the HTTP server instance and the ASF Jakarta Tomcat process, verify the following items:

- ▶ The ASF Jakarta Tomcat directory structure has the correct authorities for the user profile that will start the server. In our case, this is QTMHHTTP.
- ▶ The servlets (.class files) are located in the /TCP52D01/asfTomcatOut/TomcatHome/webapps/myapp/WEB-INF/classes directory.
- ▶ The user profile used to start the ASF Tomcat server has *READ authority to the servlets and JSP.

Now, let's see how the ASF Tomcat server works. First, you must activate the servers and then test them.

To activate the HTTP Server (powered by Apache), start the administrative GUI and follow these steps:

1. Click the **Manage** tab.
2. From the Server list, select **PBATCOUT01**.
3. Click the **Start** icon.

To activate the ASF Tomcat server:

1. From the Server list, select **Tomcat01 - ASF Tomcat**.
2. Click the **Start** icon.

To test the server, open a Web browser and enter:

`http://hostname:8401/myapp/calculator`

You should see the CalculatorExample servlet running as shown in Figure 9-22.

Calculator Example

Enter values

123 / 17 = 7.235

Calculate

Session Information

- 1. 0 + 0 = 0, Thu Jul 10 19:59:02 GMT+00:00 2003
- 2. 123 / 17 = 7.235, Thu Jul 10 20:03:11 GMT+00:00 2003

Reset Session

Figure 9-22 Out-of-process Tomcat: CalculatorExample servlet in action

If you experience any problems running the application, see Chapter 13, “Problem determination: When things do not go as planned” on page 323.

For more information

When your application has many JSP and servlets files, consider packaging them in a WAR file. Then you use the ASF Jakarta Tomcat out-of-process configuration to include a new URL path to point the WAR file.

For additional information about WAR files, go to the following Web site and enter WAR for the search criteria:

<http://java.sun.com/products/>

For additional information about how to include the WAR file in the ASF Tomcat servlet engine, see the iSeries HTTP documentation center at:

<http://www.ibm.com/eserver/iseries/software/http/docs/doc.htm>



Getting the best performance from HTTP Server (powered by Apache)

The HTTP Server (powered by Apache) provides excellent performance for Web serving. The iSeries server has held the number three spot on the SPECweb99 benchmark and the number two spot on the SPECweb99 Secure Sockets Layer (SSL) benchmark. This is a significant validation of the overall systems performance of the entire iSeries server. It is the iSeries' balanced ability to scale and run enormous On Demand Business workloads that is the basis for these (and other) benchmark successes. Of course benchmark rankings should be only one of the considerations for selecting a server.

Our ability to run enormous On Demand Business workloads is due to the integration of the SSL and Transport Layer Security (TLS) component 5722-AC3 and the HTTP Server (powered by Apache) integration with OS/400. It is also a result of the pure power of the iSeries' 64-bit RISC POWER™ processors, which allow the iSeries to climb near the top of these benchmarks.

SPECweb99 is a registered trademark of the Standard Performance Evaluation Corporation (SPEC). For details, see:

- ▶ <http://www.specbench.org/osg/web99/>
- ▶ <http://www.specbench.org/osg/web99ssl/>

Tip: The redbook *AS/400 HTTP Server Performance and Capacity Planning*, SG24-5645, is based upon the HTTP Server (original) (not the HTTP Server (powered by Apache)) and V4R4 of OS/400. Yet, this redbook is still very useful because it examines the wider integration of the HTTP server with OS/400. After all, OS/400 work management has not changed all that much since V4R4, and an HTTP server is just a “fancy file server”.

Performance in a Web server environment is influenced by many components. Understanding the components can help you to react quickly when a performance problem occurs at a crucial time. It can also help you define what exactly you can expect from your iSeries server and from your environment.

Several factors can be out of your control, such as network traffic (Internet or intranet), router capacity, client speeds, and so on, that influence the overall performance environment.

This chapter applies to companies that plan to use the HTTP Server (powered by Apache) on the iSeries server and who, from the beginning, want to tune their Web server using the correct features and components for their environment.

There are three major components of a Web server environment as shown in Figure 10-1. Each has its own performance requirement and limitations. The Web components identify:

- ▶ **Client:** The client with a Web browser represents the client component. Usually you do not have direct control over this component.
- ▶ **Network:** The network is where routers, proxy caching, communications components, and so on play an important role. This can represent the Internet, your own intranet, or both.
- ▶ **Server:** The iSeries server represents the server. Here, the performance of the iSeries server (hardware and OS/400), the HTTP server, and optionally the Web application server and the Web all work together to determine the overall server behavior in terms of performance. In general, Figure 9-1 on page 192 shows a high-level view of the layers that may be involved on your iSeries server.

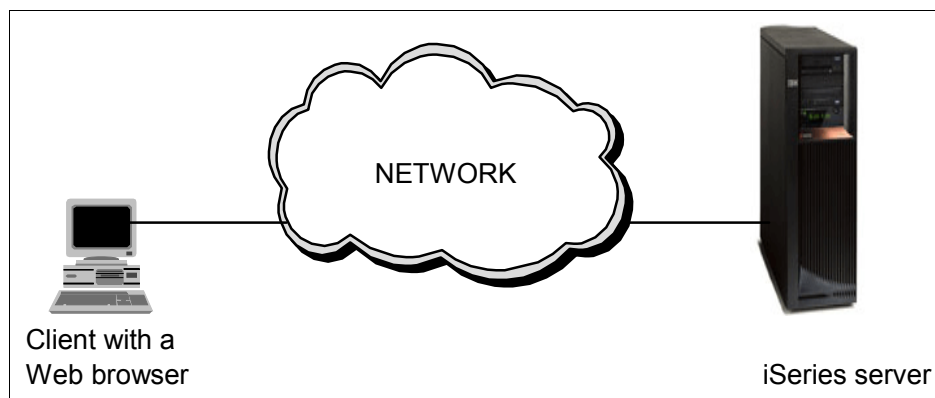


Figure 10-1 Three Web serving performance components: Client, network, and server

A problem in any of these areas may impact the performance of your Web application.

The focus in this chapter is on the iSeries server. The client and network components directly impact the Web server performance. We briefly describe each component's impact on the overall performance. The first section, 10.1, "iSeries Web server performance components" on page 226, concentrates on the iSeries server's behavior.

The client

The client typically contributes up to 25% of the end-to-end Web application response time. The client performance relies on the following resources:

- ▶ **Processor speed:** Slower clients may experience performance degradation when the Web site requires image, forms, and Java applet download and execution.
- ▶ **Memory:** Memory is an important factor inside the client because many Web-related tasks use large amounts of memory to complete. If the client does not have enough memory, the user may perceive performance problems because of the client configuration.
- ▶ **Hardware:** Each piece of hardware is important. Hard disk and communication adapters are important when performance is an issue. Keep in mind that clients are not updated as fast as IT technologies change.

- **Browser:** Since Web browsers are the main interface in Web server, you must update browsers frequently. Some Web servers, customer applications, and Web application servers rely on browser capabilities. Most of the time you need to update browser levels due to security vulnerabilities.

The network

Usually the network has more impact on overall performance than other components. The network usually contributes up to 50% of the total response time. This is because a wide variety of factors such as network traffic, bandwidth, and speed of communication lines. These factors can be understood in more detail by identifying some network components:

- Routers
- LAN topology
- Link speeds
- Packet filters
- Proxy and proxy caching
- Socks servers
- Compression of the data

With many components involved, you can spend a lot of time trying to gain a better network response time using tools to measure the behavior and never come up with an exact value. This is because the components involved in a network are dynamic components from which you can only expect average measurement and not exact values. Many of those components can be completely outside your zone of responsibility (for example, the Internet).

The server

Server behavior is impacted by several factors including application, resources, and database components. Each scenario has its own components. Do your best to create a Web application with the most current information technologies. The database access should be done with the most up-to-date utilities to data access. The Web application server and the Web server itself should be configured using the best performance practices. Figure 10-2 shows server components involved in most of the On Demand Business implementations.

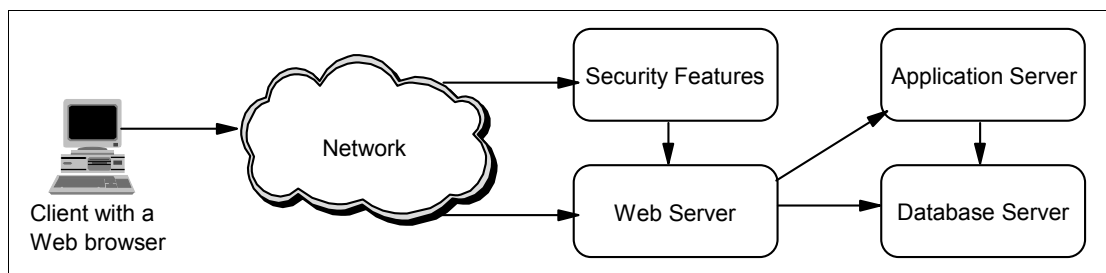


Figure 10-2 Web server components

All of these components can impact server performance, because most of the time, a client request requires processing in each one of those components.

You must analyze other iSeries internal features (for example memory, bus, and disk) for your Web application too. If the iSeries server itself does not have the internal resources to handle the requirements, performance will be impacted. Other applications running on the same iSeries can affect overall system performance too and you must take them into account.

Tip: Unique to the iSeries, HTTP server statistics are being saved into collection services in V5R2. The advantage on the iSeries server is that these reports give you a more holistic view of system performance. For more information, see 13.2.6, “Collection Services performance data” on page 345.

10.1 iSeries Web server performance components

Performance of your HTTP Server (powered by Apache) is not defined by a single “silver bullet”. It is composed of a series of configuration and design options that all work together. Figure 10-3 presents an overview of the many components that can affect the overall performance of your Web application.

If you are creating a Web application that can have many active SSL or TLS encrypted sessions starting and stopping per unit time, this can be a drain on your main iSeries PowerPC® processor or processors. As an example, most e-commerce Web sites and client-banking applications have this characteristic. It is specifically for this asymmetric public and private key exchange that IBM provides some extremely powerful hardware cryptographic adapters for you to use on the iSeries server (and other IBM @server platforms). For more information, see 10.7, “Cryptographic coprocessors” on page 300.

In Figure 10-3, you immediately notice that Fast Response Cache Accelerator (FRCA) is a task that runs completely below OS/400’s Machine Interface (MI). This allows FRCA to perform efficiently as a System Licensed Internal Code (SLIC) task, which avoids the costly overhead of switching to a user-level server thread such as the HTTP Server (powered by Apache). FRCA can serve Web content either in the form of a local cache for static content or in the form of a reverse proxy cache for dynamic content. The effect is an extremely powerful “HTTP aware” cache running in SLIC. For more information about FRCA, see 10.6, “Fast Response Cache Accelerator” on page 281.

The Network File Cache (NFC) is another component of OS/400 that was introduced with V5R2. The NFC provides the capability to efficiently store and retrieve cached files. It is like a mini-file system (open, read, write, close) for SLIC tasks and is directly used by FRCA. See “Network File Cache” on page 287 for information about how to configure the NFC for FRCA.

Using the HTTP Server (powered by Apache), you can improve the Web server performance at two different levels:

- ▶ Using global parameters that allow you to configure the attributes used by all the HTTP servers in your iSeries server. See 10.2, “Web server: Global performance values” on page 227.
- ▶ Using specific parameters based on the type of data the client is requesting. These specific parameters generally revolve around the concept configuration directives that are place in specific contexts (containers) to provide performance benefits for all files or a type of file. One example is local caches. Another is using `mod_deflate` to compress the data before it is sent across the Transmission Control Protocol/Internet Protocol (TCP/IP) network. See 10.3, “Web server: Specific performance values” on page 235.

An HTTP server, by itself, is nothing more than a fancy file server. In an On Demand Business, however, your HTTP server becomes the focal point for all Web transactions. Often dynamic content is requested of and served from a Web application. And, to avoid the costly consumption of Central Processing Unit (CPU) and memory, these Web applications often maintain their own application cache.

The Triggered Cache Manager (TCM) server is not a cache, but, like the name suggests, a cache manager. The role of the TCM is to wait for programmatic triggers (most likely as the result of an update in your Line of Business (LOB) database), which indicate that one or more Web pages, that are dynamically dependent on that updated data, must be updated. TCM then proactively regenerates those Web pages and places them in the iSeries integrated file system (IFS) (which can be thought of as a local cache for TCM) to be served as a local static file. Until the raw data in the LOB database changes again, the dynamic content is served at static file speeds by the iSeries HTTP server. The important point is that TCM only needs to update the content of the IFS if and when the raw data is updated in the LOB database. See 10.5, “Triggered Cache Manager” on page 259, for more information about this component of IBM HTTP Server for iSeries.

Also directly related to the performance of a Web server is its scalability. That is, your Web application’s ability to handle large volumes of Web traffic. To this end, the iSeries server provides a set of application programming interfaces (APIs). See Chapter 14, “High availability” on page 355.

Figure 10-3 shows the Web server components that are available to improve performance using the HTTP Server (powered by Apache).

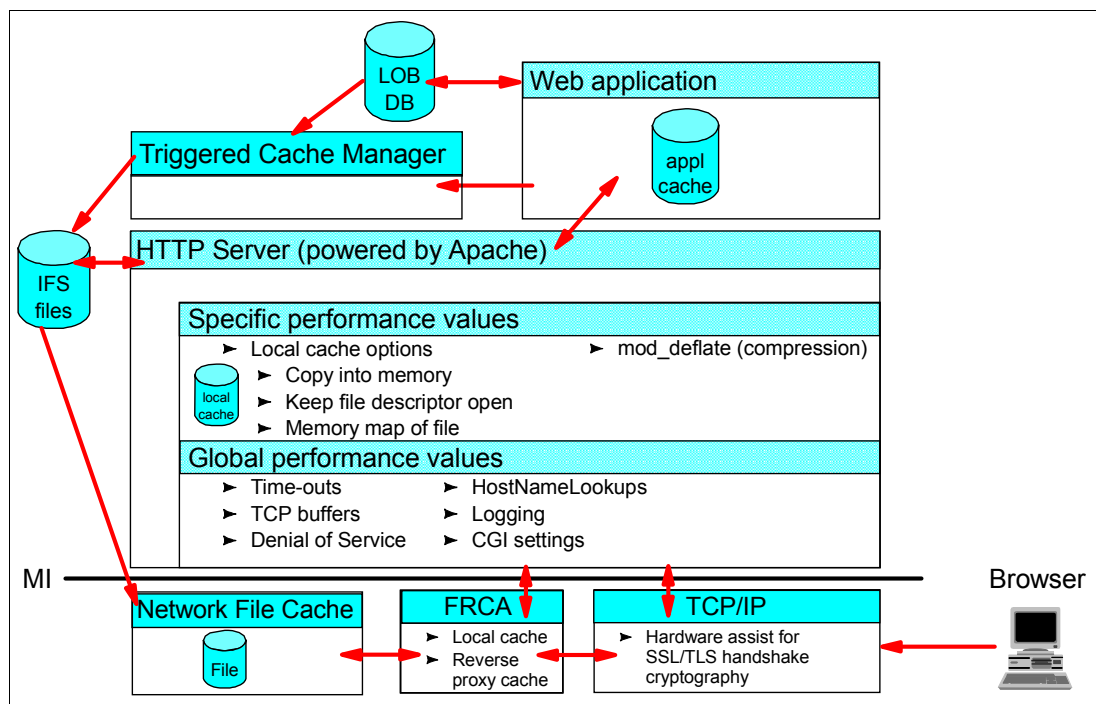


Figure 10-3 iSeries Web server performance components

Another feature that was introduced to OS/400 V5R2 and i5/OS V5R3 is the Real Time Server Statistics feature. It can help you tune the HTTP server by providing real-time information, such as number of transactions, cache utilization, and so on. You can find details about this function in 10.8, “Real Time Server Statistics” on page 301.

10.2 Web server: Global performance values

Global parameters determine the behavior of the Web server in general. These parameters are checked each time the server receives a client request. Some of these global parameters directly impact your Web server performance. Others are attributes of the Web server itself.

Each time the Web server receives a client request, the setting of these global configuration parameters can affect how the request is processed.

10.2.1 Threads and asynchronous I/O

The HTTP Server (powered by Apache) has its own multi-process model. Each HTTP server starts two (or three) processes under the QHTTPSVR subsystem:

- ▶ The manager process
- ▶ The primary process
- ▶ The backup process, when configured with the HotBackup directive

Each child process maintains its own thread pool independently.

This setting is one of the most important attributes of the HTTP Server (powered by Apache). This setting allows you to specify how many threads each child process is allowed to use. The default value is the same as the value for the maximum number of threads found on the Global Server Settings form. The directive is *ThreadsPerChild*.

That is, you can set this parameter at the Global Server Setting to be the default value at startup for all your Web servers as shown in Figure 10-4. Then you have the option to override this Global Server Setting for each HTTP Server (powered by Apache).

You can *only* configure the maximum number of threads that the server opens at startup. The HTTP Server (powered by Apache) always starts with the maximum number of configured threads.

When no threads are available, the Web server response time, from the client point of view, is impacted since the request takes longer because of the lack of available threads. Setting this number too low impacts the server performance since the client request cannot be processed until the Web server finds an available thread. But setting the maximum number of threads too high, in general, requires more system resources to keep those threads available for use. There is no optimum value for this setting.

With the HTTP Server (powered by Apache) implementation, the HTTP Server processes communications requests asynchronously. In this asynchronous input/output (I/O) model, threads are only involved in processing when work is to be done. Threads are dispatched to perform work as required. When not performing work, the threads are returned to a pool of available threads making the server process more efficient and improving performance by using the thread resources better. Asynchronous I/O also makes the server more scalable to support a high number of users especially when combined with persistent connections. We recommend that you keep the default value of *on* (or enabled). The directive is *AsyncIO*.

Tip: Asynchronous I/O is one of many enhancements to the standard Apache server as delivered to IBM Rochester by the Apache Software Foundation (ASF). This is just one of the many reasons that the parenthetical phrase (powered by Apache) means integration.

Setting the maximum number of threads for all HTTP servers

This parameter is configured on the Global Server Settings display for all your HTTP servers as shown in Figure 10-4. To set the maximum number of threads, follow these steps:

1. From the IBM Web Administration for iSeries interface, select the **Advanced** tab and then the **Settings** subtab.
2. In the left pane, under Global Settings, select **Global Server Settings**.

3. In the right panel, for Number of threads, type the maximum number of threads in the Maximum field.

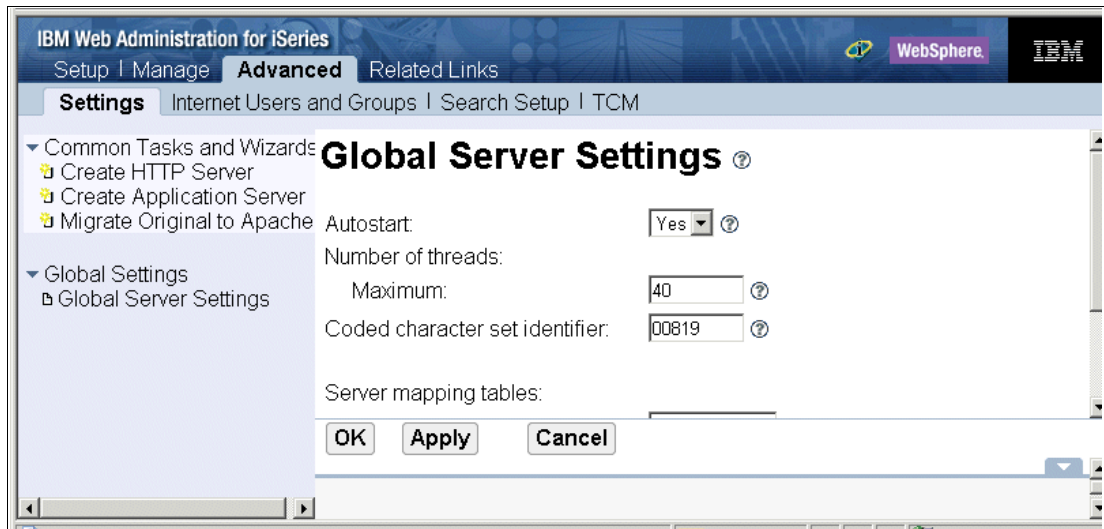


Figure 10-4 Global Server Settings: Setting the maximum number of threads for all HTTP servers

Setting the maximum number of threads for a single HTTP server

You can also set the maximum number of threads and the option to use asynchronous I/O for each individual HTTP Server (powered by Apache) as shown in Figure 10-5. Follow these steps:

1. Select the **Manage** tab.
2. Make sure you select the server you want to manage. For Server area, select **Global configuration**.
3. In the left pane, under Server Properties, select **System Resources**.
4. Click the **Advanced** tab.
5. In the right panel, change the Number of threads to process requests to override the Global Server Settings. Click **OK**.

10.2.2 Process control: HotBackup

The HotBackup directive is used to specify whether a hot backup server should be started at server startup time. With the hot backup server active, if the primary server job abnormally terminates, the hot backup immediately takes over, acts as the primary, and continues servicing requests. A new hot backup is automatically created, in the background, within one minute.

If the primary server process failure is not due to the network, all user connections remain active during the hot backup takeover and the end users do not detect the loss of a server. However, some HTTP requests in transient may be lost. If the failure is due to the loss of network, the server must be restarted. With HotBackup off, only one multi-threaded server child process is started.

Tip: An example of a loss of network may be if one of two interfaces on the iSeries fail. The routes bound to the failing interface cause all the connections across that interface to fail. A good solution to this potential problem is to configure a *virtual Internet Protocol (IP) address* (or a circuitless connection) that is an IP interface defined on the system without being associated with a physical hardware adapter. These addresses can always be active on the system. For example, if one of two physical interfaces fail, then all network traffic can be re-routed through the active interface. The applications and HTTP server will never know of the problem. For more information about using virtual IP addresses, see *iSeries IP Networks: Dynamic!*, SG24-6718.

You can configure this setting as explained here and shown in Figure 10-5:

1. Select the **Manage** tab.
2. For Server, select the server you want to manage. For Server area, select **Global configuration**.
3. In the left pane, under Server Properties, select **System Resources**.
4. Click the **Advanced** tab.
5. On the Advanced page, select the **Initialize a backup process to take over in the event of server process failure** option. Click **OK**.

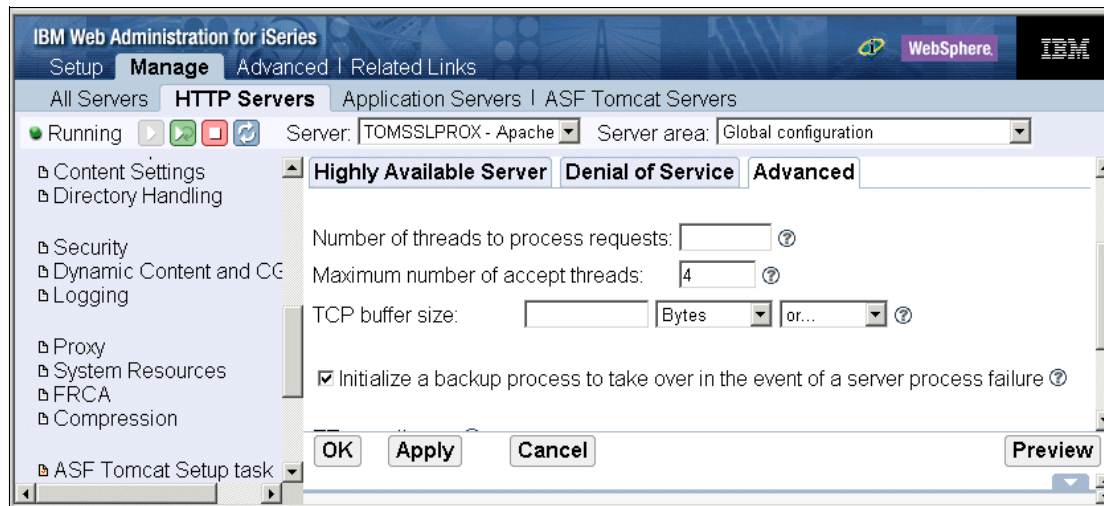


Figure 10-5 HotBackup and threads configuration

You can also override the default value found in the Global Server Settings form for the directive `ThreadsPerChild` by specifying the number of threads that each child process is allowed to use.

10.2.3 Logging

Logging is another setting that impacts server performance. Simply stated, as you request a higher logging level, a greater load is placed on the server to write more information in the log file. For example, if the logging level is set to *Debug* and the Web server experiences a problem, messages written to the error log file increase and the Web server performance may decrease.

You can configure this parameter for every HTTP server and for each virtual context within the HTTP server. If the HTTP server has a different Error Log file for every virtual host context, consider that a file descriptor is opened for each log file. Opening too many descriptors can impact system performance.

For more information about logging with the HTTP Server (powered by Apache), see 13.2.3, “Server logs” on page 331.

10.2.4 HostNameLookups

The HostNameLookups directive enables Domain Name System (DNS) lookups so the host names can be logged (and passed to Common Gateway Interface (CGI) and server-side includes (SSI) in the REMOTE_HOST environment variable). That is, it causes your HTTP server to do a reverse lookup to convert an IP address into a host name and domain. This may make it easier to track the usage of your Web site (by geography, for example) or to determine problems.

The default for this directive is *off* to save on network traffic for those sites that do not truly need the reverse lookup. Heavily loaded sites should leave this directive set to *off*, since DNS lookups can take a considerable amount of time and resource.

You can configure this setting using the HostNameLookups directive. To configure this directive, in the left pane under Server Properties, click **General Server Configuration**. Then in the right panel, click the **General Settings** tab.

10.2.5 KeepAliveTimeout

The KeepAliveTimeout setting is used to control whether the Web server works with persistent connections. Persistent connections enable a single TCP connection to be used for multiple HTTP requests. Normally, each HTTP request is made over a separate connection. Reusing a connection reduces the overhead, thereby improving performance for that client.

When the server runs with persistent connections, the KeepAliveTimeout setting determines the number of seconds that the server waits for subsequent requests before closing the connection. If this value is too low, the server can be impacted in terms of performance since connections can close frequently. If this value is too high, the Web server can have many connections open and the server can run out of resources. In this case, using asynchronous I/O can alleviate (but not eliminate) the problem of running out of resources.

Tip: When migrating an HTTP Server (original) instance to an HTTP Server (powered by Apache) instance, the value of the KeepAliveTimeout is set to 4 seconds by the migration utility. This may cause problems for many environments. We recommend that you set the value to 300 seconds after the migration completed.

You can configure this global setting as explained here (see Figure 10-6):

1. Select the **Manage** tab.
2. For Server, select the server you want to manage. For Server area, select **Global configuration**.
3. In the left pane, under Server Properties, select **System Resources**.
4. In the right panel, click the **HTTP Connections** tab.

5. All the parameters on the HTTP Connections page (Figure 10-6) relate to KeepAliveTimeout. Click **OK**.

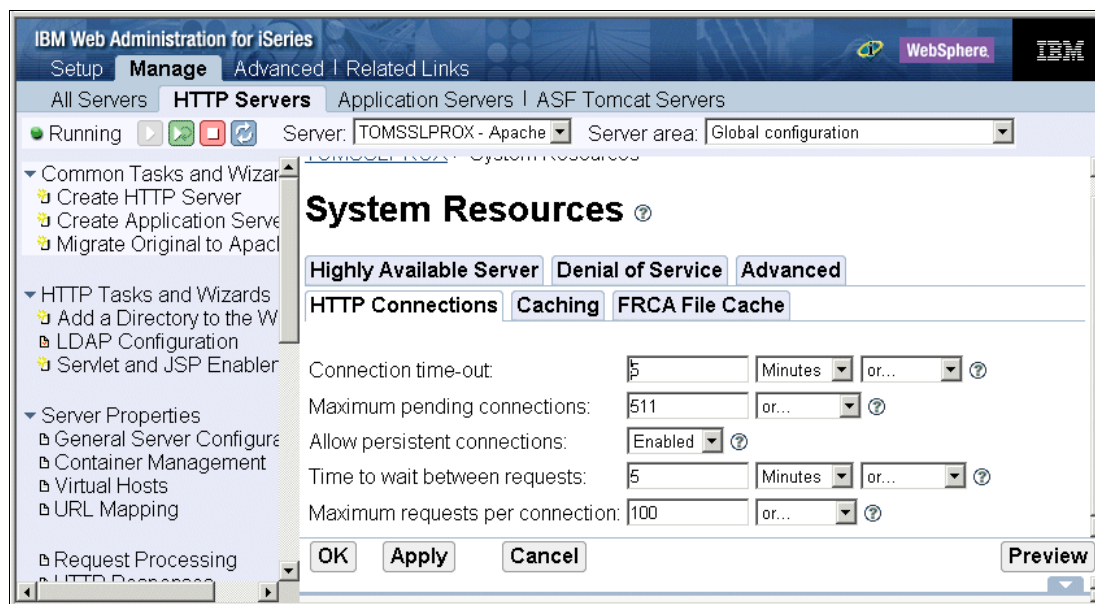


Figure 10-6 KeepAliveTimeout directive

This value applies to each client request. If the Web server you are working with is used on the Internet, keep in mind that, since this is a communication setting, the value you select here can be correct for some environments but not for others. We recommend that you leave the default value unless you know your environment and have reason to make a change. One reason may be that you know that persistent connections are not supported all the way between the client. And you also know that your server or the majority of browsers that connect to your site do not support persistent connections.

10.2.6 TCP buffer size

The TCP buffer size attribute provides a limit on the number of outgoing bytes that are buffered by TCP. After this limit is reached, attempts to send additional bytes may result in the application blocking until the number of outgoing buffered bytes drops below this limit causing a negative impact in the Web server performance. The default value for this setting is zero. This means the Web server uses the TCP value configured in the iSeries server for the TCP send buffer size (TCPSNDBUF) parameter in the Change TCP/IP Attributes (CHGTCPA) command. The default value for TCPSNDBUF is 102400.

You can configure this setting as explained here (see Figure 10-5 on page 230):

1. Select the **Manage** tab.
2. For Server, select the server you want to manage. For Server area, select **Global configuration**.
3. In the left pane, under Server Properties, select **System Resources**.
4. In the right panel, click the **Advanced** tab.
5. In the TCP buffer size field, type the TCP buffer size. The value for the TCP buffer size is best left as the default of 0, unless application-specific reasons exist for your server. Click **OK**.

10.2.7 Denial of service

The denial of service attribute is equally both a performance setting and a security setting. This setting allows you to identify the possibility of an attack based on the data frame size. The HTTP server may identify an attack because the frame size differs from the one it expects. Although this setting impacts the server performance as each request is tracked, it allows you to prevent a more dangerous performance degradation when dealing with a type of attack that may intentionally slow down or even completely paralyze your server. The HTTP Server (powered by Apache) includes the following attributes to prevent a denial of service attack:

- ▶ **Maximum message body size:** Allows you to limit the size of an HTTP request message body within the context the directive is given (server, per-directory, per-file, or per-location). The default value is zero, which indicates that no maximum is size specified. The directive is *LimitRequestBody*.
- ▶ **Maximum XML message body size:** Allows you to limit the size of an Extensible Markup Language (XML)-based request body. The default value is 1000000 bytes. The directive is *LimitXMLRequestBody*.
- ▶ **Maximum header fields:** Allows you to modify the limit on the number of request header fields allowed in an HTTP request. The default value is 100. The directive is *LimitRequestFields*.
- ▶ **Maximum header field size:** Allows you to limit the size for an HTTP request header field below the default size compiled with the server. The default value is 8190. The directive is *LimitRequestFieldSize*.
- ▶ **Maximum HTTP request-line:** Allows you to limit the size for a client's HTTP request line below the default size compiled with the server. The default value is 8190. The directive is *LimitRequestLine*.

You can configure the denial of service settings as explained here (see Figure 10-7):

1. Select the **Manage** tab.
2. For Server, select the server you want to manage. For Server area, select **Global configuration**.
3. In the left pane, under Server Properties, select **System Resources**.
4. Click the **Denial of Service** tab.

5. On the Denial of Service page (Figure 10-7), enter the values for your environment. Click **OK**.

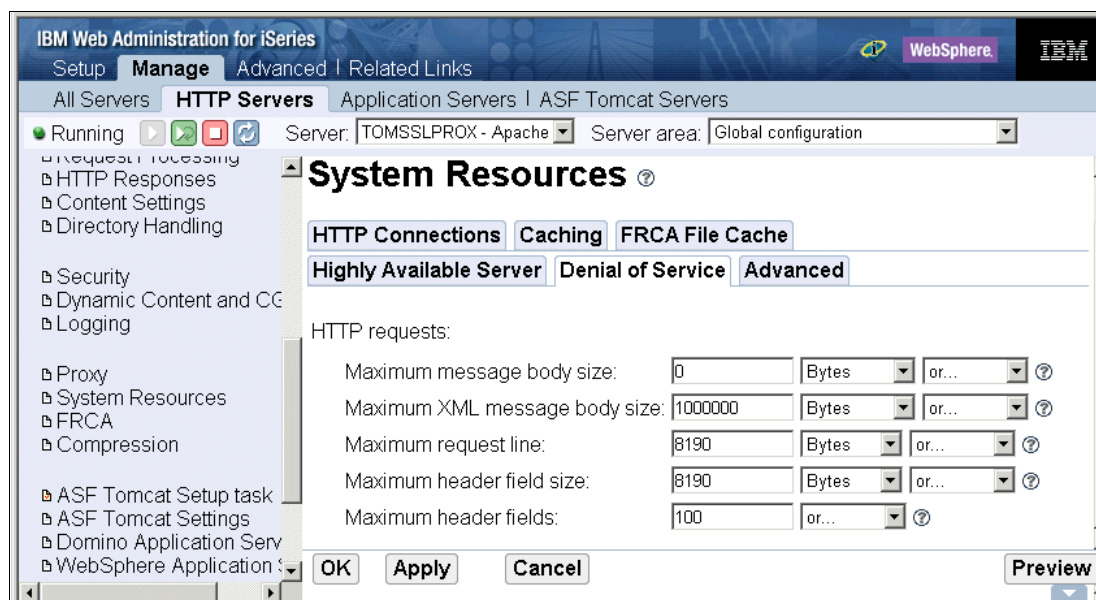


Figure 10-7 Denial of Service page

See Chapter 6, “Defending the IFS” on page 101, for additional information.

Tip: This built-in protection against various denial of service attacks is one of the many integrated extensions in the standard Apache Web server. Again, it is one of the reasons for the (powered by Apache) parenthetical phrase in the HTTP Server (powered by Apache) formal name of the server on the iSeries server.

10.2.8 CGI initialization at server startup

CGI initialization Uniform Resource Locator (URL) support offers the capability to start and initialize CGI programs, such as Net.Data or other CGI programs, at server startup time. This can significantly improve performance.

To activate this feature, you have to define prestarted CGI helper jobs (StartCGI directive) with a corresponding user under which the job will run. Then you must add entries for the particular CGI programs (CgiInitialUrl directive) that should be started. The entry consists of the fully qualified physical path of the CGI program along with the user ID. The user ID must refer to an entry in the CGI helper job section. The server does not start if the directive CgiInitialUrl is configured without the directive StartCgi. You can set the same definitions for threaded CGI jobs.

Perform the following steps to configure CGI initialization at server startup time.

1. Select the **Manage** tab from the IBM Web Administration for iSeries interface.
2. For Server, select the server that you want to manage. For Server area, select **Global configuration**.
3. In the left pane, under Server Properties, select **Dynamic Content and CGI**.
4. Click the **Server Startup** tab.

5. On the Server Startup page (Figure 10-8), complete these tasks:
 - a. Under Prestarted CGI helper jobs, click **Add**.
 - b. Select the type of job and enter the number of jobs to be pre-started as well as the user under which the jobs run. Click **Continue** to add this entry.
 - c. Under the Prestarted CGI programs section, click **Add**.
 - d. Enter the path of the CGI program in IFS naming format and select the user under which the program will be run. Click **Continue**.
 - e. Click **OK** to save the new configuration.

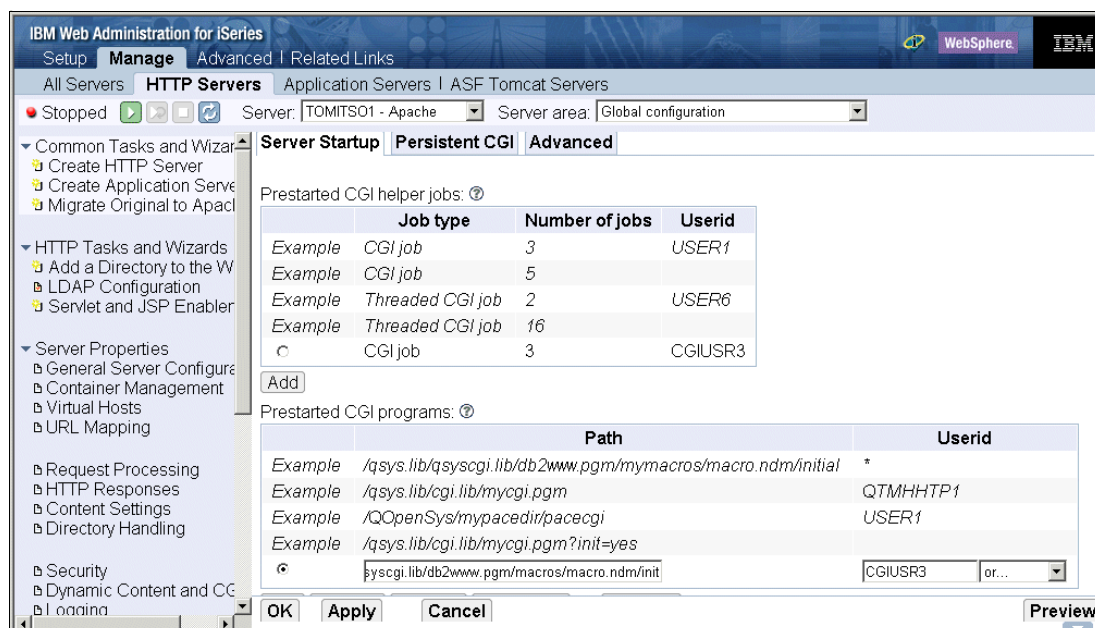


Figure 10-8 Dynamic Content and CGI: Server Startup tab

6. Restart the server to activate the changes.

10.3 Web server: Specific performance values

Specific performance values are the settings that you can use to improve Web server performance based on the type of data the server is going to serve. From a lifetime point of view, all data is dynamic. It is just that some data is more dynamic than others.

To illustrate this point, you may declare that your home page is static, considering it is made up of Hypertext Markup Language (HTML), Java script, and GIFs. Even your home page changes from time to time. When it does, you want it (and all the other popular places in your Web site) to be cached for the best performance. All Web content is dynamic, in this sense.

To ease our understanding, we define content that does not change that often as *static*. Content that changes based on database information and user input is called *dynamic*.

In the end, however, the best way any Web server can improve its performance is by caching the content *before* it is requested. For this purpose, the HTTP Server (powered by Apache) supports these caching mechanisms:

- ▶ HTTP Server (powered by Apache) local cache
- ▶ HTTP Server (powered by Apache) proxy cache

- Fast Response Cache Accelerator
- Triggered Cache Manager

Tip: Recall that TCM is not a cache but a cache manager. In effect, the TCM helps you to be proactive in the update of HTML Web pages that you traditionally thought to be dynamic and place them where your HTTP Server (powered by Apache) serves them from the IFS as static content.

The HTTP Server (powered by Apache) local cache mechanism is generally used with static data. TCM is used to improve performance for dynamic data. FRCA, on the other hand, can be used to cache both static and dynamic contents. For a review of the components of performance, including the relationship between the HTTP and TCM servers, FRCA, and the many caches used by a Web application, see Figure 10-3 on page 227.

In addition to caching, some types of files can be compressed by your HTTP Server (powered by Apache) before you send through the TCP/IP network. By compressing the files, you are both allowing more information to be carried by the same size network and, in many cases, allowing a single transaction to arrive faster. The downside is additional CPU and memory requirements on both the server and client. For more information, see 10.4, “Increasing throughput with compression” on page 240.

10.3.1 HTTP Server (powered by Apache) local cache

The local cache is used to cache data in system memory that is more static in nature. Static, in this case, means the content is not changing based on the user input or database information. In general, static content includes image files, HTML pages, etc. By keeping this data loaded in the server’s memory, you can improve server response time for files because the server can handle the request far more quickly than if it had to read from the file system. The HTTP Server (powered by Apache) allows you to configure which files will be preloaded in the server’s memory at server startup and the amount of memory used for this purpose.

The local cache implementation is a two-stage process:

1. Define the memory size for the files to be cached.
2. Define the cache method.

The local cache implementation uses one main storage space for all the local cache files, so the memory size you define is used for all the cache files. This includes both the files that are cached at server startup time and any changed or new files cached due to dynamic caching (see “What to cache?” on page 237). The server directive to identify the memory size is *CacheLocalSizeLimit*.

Files can be cached at server startup using any of these three methods:

- Copy into memory
- Keep file descriptor open
- Memory map of file

Tip: By default, the QHTTPSVR subsystem runs in system pool 1. Memory allocated by the *CacheLocalSizeLimit* directive is from this pool. To verify which pool is used by the QHTTPSVR subsystem, enter the 5250 Display Subsystem Description (DSPSBSD) command:

```
DSPSBSD SBSD(QHTTPSVR/QHTTPSVR) OUTPUT(*)
```

Enter option 2 (Pool definitions).

Copy into memory

The copy into memory method allows you to define the file or files that will be pre-loaded in the iSeries memory, in the memory pool used by the HTTP server at Web server start up. The memory size can be set up according to your application requirements. There is no limit for the memory size used to preload the files. The limitation relies on the iSeries memory capabilities.

The server directive used to cache files using this method is *CacheLocalFile*.

Keep file descriptor open

The Keep the file descriptor open method allows you to define ASCII file or files whose descriptors are cached at server startup. Here, files are not copied into memory (they do not allocate large amount of memory) and yet provide similar performance. Files cached with this method remain open, shared read, while the server is active.

The server directive used to cache files using this method is *CacheLocalFD*.

Memory map of file

The memory map of the file method is similar to the copy into memory method. The difference here is that this method uses a memory pointer to specify files that should be cached at startup, which means that files are not copied into memory.

The server directive used to cache files using this method is *CacheLocalFileMmap*.

What to cache?

A powerful pair of options gives your HTTP Server (powered by Apache) server the ability to:

Tip: If the file is updated, then the local cached entry for this file is marked invalid and the file is served from the IFS for all subsequent requests. Only restarting your HTTP Server (powered by Apache) causes the file to be placed back into the local cache.

FRCA local cache has an interesting feature. Assume that a file located in the IFS is cached in the NFC by FRCA. If that file is updated in the IFS, it is also automatically updated in the NFC and the new content is served by FRCA. For details, see 10.6, “Fast Response Cache Accelerator” on page 281.

- Dynamically update the (static) files that were placed in the local cache at server startup time. The default value is on (or enabled).

This directive (*LiveLocalCache*) checks to see if the file is updated in the IFS each time it is requested. If it is not updated, the file is served from the cache. If it is updated, then the entry for this file in the local cache is marked invalid and the file is served from the IFS for all subsequent requests. You restart your server to load it back in the local cache.

If *LiveLocalCache* is off, then your HTTP Server (powered by Apache) server does not check to see whether the file has changed in the IFS.

Clearly, *LiveLocalCache off* gives you the best performance for your Web server at the expense of denying you the ability to update a particular file. *LiveLocalCache off* is useful in a directory of all GIFs, for example, that rarely change.

- Dynamically add new (static) files to the local cache based on demand. The default value is off (or disabled).

This directive (DynamicCache) allows dynamic caching of (static) files. There is overhead involved in determining whether a file being served should be added to the cache that impacts *all* the files being served from your Web server. Because of this, we recommend that you use this directive for sites that generally have less than 1000 files.

Or, to put this another way, if you have less than a 1000 static files to serve and you have not done an analysis as to which files to populate your local cache at server startup time, then you may try using DynamicCache *on*. But, it is always better to identify all the files you want to add to the local cache at server startup time since this is far more efficient.

The dynamic cache only adds files to the local cache as long as there is still room as defined by the directive CacheLocalSizeLimit. If the local cache is full, no more files are added to the cache.

Example configuration of a local cache

The cache file methods are usually configured through the Administration graphical user interface (GUI). To configure, for example, all the GIF files in your document root to be cached at server startup, follow the steps as explained here and shown in Figure 10-9.

1. Start the IBM Web Administration for iSeries interface, and select the **Manage** tab.
2. From the Server list, select the server you want to manage. From the Server area list, select **Global configuration**.
3. In the left pane, under Server Properties, select **System Resources**.
4. Click the **Caching** tab.
5. On the Caching page, complete these steps:
 - a. In the Maximum storage size field, type the memory size that you want to configure for the files to cache. Remember that this size is the main storage size used by all the files that you decide to preload into memory. For this example, we use the default 2000.
 - b. Under the Files to cache when server is started table, click **Add**.
 - c. In the File path and name column, enter the file path for the static data the Web server will cache at server startup. For this example, we cache the images (*.gif) files of the document root /www/tomitsol/htdocs/images/*.gif.
 - d. In the Cache method column, select the mechanism used to cache the files. For this example, we select **Copy into memory**.
 - e. Click **Continue** to save the new entry to the list of files to be cached at server startup.
 - f. Click **OK**.

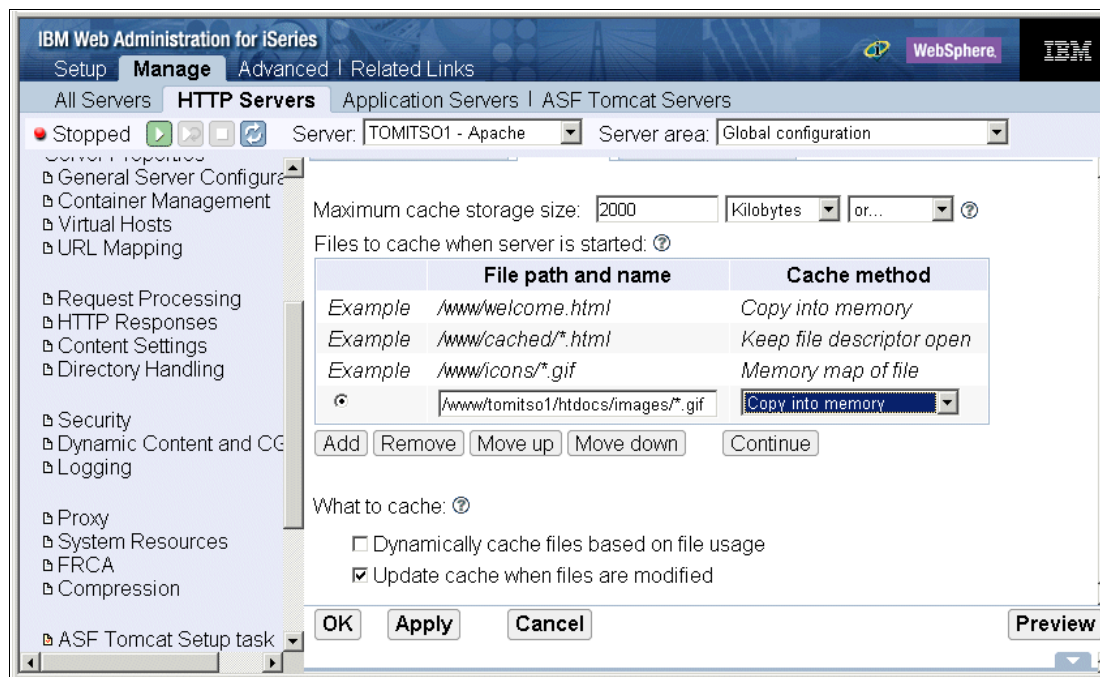


Figure 10-9 Web server cache configuration

One Web server configuration can have many cache entries, since one specific environment can require a different cache method for a group of files. Each directory may have different static files that are requested frequently. To improve the server response time of those frequently access files, we decide to preload those files in memory when the server starts.

You can configure the Web server cache entries only from the global settings attributes. You cannot configure this directive for any other context than the HTTP general context.

Using HTTP server trace to see how the local cache is populated at startup

Starting your Web server with the `-vv` (very verbose) trace (see 13.2.5, “HTTP server trace” on page 341, for details about how to start and dump the `-vv` trace information) causes the HTTP Server (powered by Apache) to document how many files are added to the local cache.

You can then search the HTTP server trace scanning for the text “cache”. At the beginning of the trace information, you see many entries reading and interpreting the cache directives found in the configuration file. Later, you see trace entries similar to this example:

```
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+
000000AB:274632 total number of cache FCs = 51, FDs = 0, FRCAs = 2, and MMAPs = 0.
000000AB:274688 total allocated cache size: LOCAL = 0 80877, FRCA = 0 17435.
```

The first line indicates that the total number of files cached by copy (FCs) was 51, file descriptors (FDs) was 0, FRCA was 2 (see 10.6, “Fast Response Cache Accelerator” on page 281), and files cached as memory mapped (MMAPs) was 0. The second line indicates the total local cache memory consumed by these cached items is 80,877 bytes.

10.3.2 HTTP Server (powered by Apache) proxy cache

When the HTTP Server (powered by Apache) is acting as either a forward proxy or reverse proxy it can cache the results of the content received from the remote HTTP server. This can provide significant performance benefits for multiple Web clients who are all requesting the same document.

See 6.5, “Proxy server: Protecting direct access” on page 142, for more information about both the forward and reverse proxy configurations. You should also see 10.6, “Fast Response Cache Accelerator” on page 281, which demonstrates another method to configure a reverse proxy cache.

10.4 Increasing throughput with compression

Data compression is another mechanism that can be used to improve performance. The HTTP server module that provides compression support is the `mod_deflate` module. It is a powerful module that allows you to compress, by configuration, files that are served from your HTTP Server (powered by Apache). `mod_deflate` is Apache’s open source equivalent to `mod_gzip`. Compressing the data before it is sent by your HTTP Server (powered by Apache) can dramatically save on network delays caused by bandwidth restrictions. The data is uncompressed at the remote client’s Web browser. `mod_deflate` is useful in networks where individual network links are saturated with traffic or the end user is connected via modem.

As an anecdotal example, the `/index.html` home page that is served from our NetObjects Fusion generated sample Web application that we use in this redbook is compressed by `mod_deflate` from 10867 to 2002 bytes. Another example shows the HTML output of a `Net.Data` macro is compressed from 10631 to 2869 bytes. Some files do not compress as well. Examples are JPEGs, PDFs, and files that are already compressed. `mod_deflate` allows you to configure which types of files are compressed and which are not.

Also, some documents are not supported by browsers in compressed form, such as JavaScript (`.js`). In such cases, they must be excluded. They may work in the future, but for now, it can make more problems than benefits.

And, of course, nothing is for free. This processing takes additional CPU and memory on your iSeries server as well as the remote client browser. You must determine if the savings in the size of the documents that you are sending through the network outweigh the performance impact on the server and client.

Compression was initially added to 5722-DG1 at V5R1 and V5R2 via PTFs. One PTF contains `zlib` and the other PTF contains the `mod_deflate` module plug-in:

- ▶ V5R1: PTFs SI09287 and SI09223
- ▶ V5R2: PTFs SI09286 and SI09224

Initial compression support did not support the configuration via the administration GUI. However, the GUI configuration support was added with the following group PTFs for V5R1 and V5R2.

- ▶ V5R1 with group PTF SF99156 level 17 (December 2003)
- ▶ V5R2 with group PTF SF99098 level 13 (December 2003)

The GUI configuration support for V5R3 is implemented in the base code.

You can find more information about `zlib` on the Web at the following site, which announces that `zlib` is “A Massively Spiffy Yet Delicately Unobtrusive Compression Library”:

<http://www.gzip.org/zlib/>

Important: `zlib` support is provided as a service program (`*SRVPGM`) in `QHTTPSVR/QZLIBZLIB`. Just as `mod_deflate` uses this service program to compress data, your applications can too. The IBM Rochester lab brought `zlib` to the iSeries to support `mod_deflate` and not *your* application. That is, IBM does not support use of this service program.

10.4.1 Compression considerations

You can configure compression individually for input and output traffic. Filters are used to define the kind of traffic that should be compressed or decompressed. The only valid filter name that is provided by the HTTP Server (powered by Apache) is DEFLATE. The DEFLATE filter uses gzip for compression. However, you can write your own compression utilities that can be plugged into the HTTP server configuration. These compression utilities are accessible by a filter name of your choice.

Typically, you set up your HTTP server to compress only outbound traffic, because the amount of data sent to a browser is much higher than the request data received from a browser. The benefit of reducing the size of data sent from the server to the browser outweighs the overhead that compression introduces for the CPU. One reason not to use `mod_deflate` for inbound traffic is that many browsers do not support compression of HTTP requests.

Carefully plan for the Multipurpose Internet Mail Extensions (MIME) or file types you want to compress. As indicated earlier, if you configure the server to compress all outbound traffic, the server would also compress files, such as JPEG files, that are already compressed. This can cause more overhead than performance gain.

The IBM Web Administration for iSeries interface GUI provides three configuration tabs under the category Compression. They provide the configuration options that are described in the following sections.

Input filters

Input filters can be defined for the global configuration, directory, and virtual host contexts. They determine how data received from the browser is handled. You have the option to activate decompression for all traffic received within a specific context or traffic (files) that has a certain extension. As mentioned previously, this option is not used often due to the lack of HTTP request compression support for most browsers.

Output filters

As for input filters, output filters can also be defined for the global configuration, directory, and virtual host contexts. However, output filters provide more configuration flexibility. You can enable compression for an entire context, for certain browser versions, for certain file extensions, or for certain MIME types.

Advanced

Using the Advanced tab, you can control the logging of compression information and fine-tune the compression behavior of the `mod_deflate` module. In addition, there are several more configuration tabs that can impact compression, but are not directly related to the `mod_deflate` module.

10.4.2 Example configurations

Here are three sample configurations that provide you with a simple starting point for configuration and use of `mod_deflate`:

- ▶ Compressing everything in a context
- ▶ Compressing only HTML files for specific Web browsers
- ▶ Compressing only a few MIME types in a context using `AddOutputFilterByType`

Compressing everything in a context

The first configuration example sets up the server to compress outbound traffic for all files that are served from a specific context (/www/tomits01/datasheets/). In this case, the context (IFS directory) contains only HTML and text files.

Note: If the selected directory also contain files, such as JPEG files, the CPU overhead for compression may outweigh the benefit of reduced data traffic. In this case, compress only certain types of files as described in “Compressing only a few MIME types in a context” on page 250.

1. From the IBM Web Administration for iSeries interface, click **Manage**.
2. From the Server list, select the server that you want to configure.
3. From the Server area list, select the context for which you want to configure compression.
4. In the left pane under Server Properties, click **Compression**.
5. In the Compression panel (Figure 10-10), select the **Output Filters** tab.

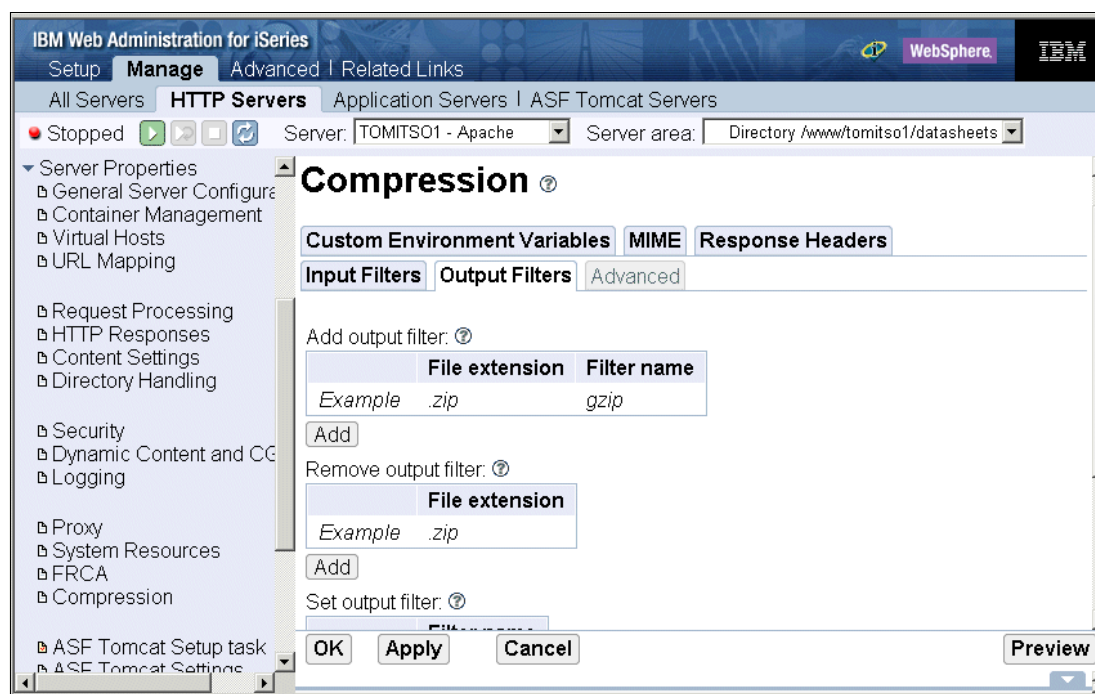


Figure 10-10 Compression: Output Filters tab

6. On the Output Filters page (Figure 10-11), complete these tasks:
 - a. Scroll down to the Set output filter section. Click **Add** to add a new entry.
 - b. Enter the filter name DEFLATE. This is the only valid filter name that you can enter and that has an impact on compression unless you have written your own compression module with your own filter name. The DEFLATE name refers to the gzip compression library.
 - c. Click **Continue**.
 - d. Click **OK** to save your settings.

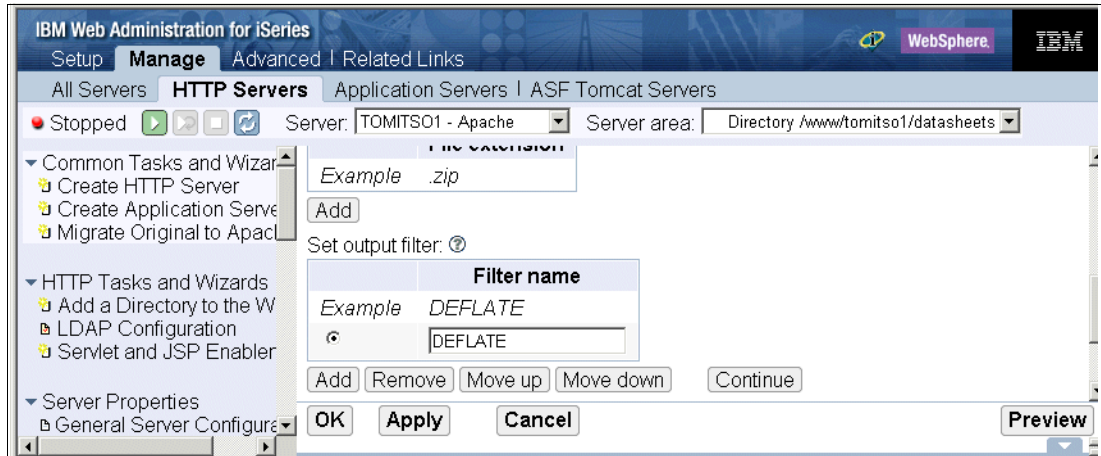


Figure 10-11 Compression: Output Filters tab Set output filter section

- Restart your server to activate the changes. From now on, all files that are served out of the context /www/tomitso1/datasheets are compressed when sent to the client.

Example 10-1 shows the key directives that are needed to support compression using the `mod_deflate` module in the `httpd.conf` file in bold. These directives were added by the IBM Web Administration for iSeries interface. Most of the other directives that do not directly affect this example were removed.

Example 10-1 mod_deflate: Minimal configuration directives

```
# Configuration originally created by Create HTTP Server wizard on Wed Sep 22 15:54:41 CEST
2004
LoadModule deflate_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
Listen *:8001
DocumentRoot /www/tomitso1/htdocs
...
<Directory />
    Order Deny,Allow
    Deny From all
</Directory>
<Directory /www/tomitso1/datasheets>
    Order Allow,Deny
    Allow From all
    SetOutputFilter DEFLATE
</Directory>
<Directory /www/tomitso1/htdocs>
    Order Allow,Deny
    Allow From all
</Directory>
Alias /datasheets/ /www/tomitso1/datasheets/
```

Tip: You use `RemoveOutputFilter DEFLATE` to turn off `mod_deflate` in a subcontext.

Compressing only HTML files for specific Web browsers

Some versions of Web browsers cannot handle the compression of `mod_deflate`. In addition, it is not always good to try to compress graphic files and other files of certain types. The following example provides a good starting example that compresses only files of type HTML for all incoming URLs as defined by the `<Location />` directive. Example 10-2 highlights the key directives that we added to a standard `httpd.conf` file. Most of the other directives that do not directly affect this example were removed.

This example uses the recommended configuration found on the Web at:

http://httpd.apache.org/docs-2.0/mod/mod_deflate.html#recommended

Tip: The configuration example at the httpd.apache.org Web site requires minor modification for it to work in the HTTP Server (powered by Apache). The original syntax at httpd.apache.org was:

```
# Don't compress images
SetEnvIfNoCase Request_URI \
\.(?:gif|jpe?g|png)$ no-gzip dont-vary
```

We modified the SetEnvIfNoCase directive to this one line syntax (shown in Example 10-2):

```
# Do not compress images
SetEnvIfNoCase Request_URI \.(gif|jpe?g|png)$ no-gzip dont-vary
```

Example 10-2 mod_deflate: Recommended configuration directives from ASF

```
# Configuration originally created by Apache Setup Wizard Wed Apr 30 15:38:30 UTC 2003
LoadModule deflate_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
Listen *:8000
DocumentRoot /tcp52d00/basicconfig/itsoco
ServerRoot /tcp52d00/basicconfig
...
<Location />
# Insert filter
SetOutputFilter DEFLATE
# Netscape 4.x has some problems...
BrowserMatch ^Mozilla/4 gzip-only-text/html
# Netscape 4.06-4.08 have some more problems
BrowserMatch ^Mozilla/4\.0[678] no-gzip
# MSIE masquerades as Netscape, but it is fine
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
# Do not compress images
SetEnvIfNoCase Request_URI \.(gif|jpe?g|png)$ no-gzip dont-vary
# Make sure proxies do not deliver the wrong content
Header append Vary User-Agent env=!dont-vary
</Location>
...
<Directory />
    Order Deny,Allow
    Deny From all
</Directory>
<Directory /tcp52d00/basicconfig/itsoco>
    Order Allow,Deny
    Allow From all
</Directory>
```

You can find this example at the Apache Web site. It uses BrowserMatch directives that were introduced with the mod_browser module. However, the BrowserMatch directives were declared obsolete with Apache 1.2 and higher. A replacement for the BrowserMatch directive is the SetEnvIf directive of the mod_setenvif module. The HTTP Server (powered by Apache) still accepts the BrowserMatch directive when manually configured, but the IBM Web Administration for iSeries interface supports only the SetEnvIf directive in the GUI.

The following steps explain how to implement the previous example using the IBM Web Administration for iSeries interface. This time, the directives apply to a directory context instead of a location context.

1. From the IBM Web Administration for iSeries interface, click **Manage**.
2. From the Server list, select the server you want to configure.
3. From the Server area list, select the context for which you want to configure compression.
4. In the left pane under Server Properties, click **Compression**.
5. In the Compression panel, select the **Output Filters** tab.
6. On the Output Filters page, complete these steps:
 - a. Scroll down to the section Set output filter and click **Add** to add a new entry.
 - b. Enter the filter name DEFLATE. This is the only valid filter name that you can enter and that impacts compression unless you have written your own compression module with your own filter name. The DEFLATE name refers to the gzip compression library.
 - c. Click **Continue**.
 - d. Click **OK** to save your settings.

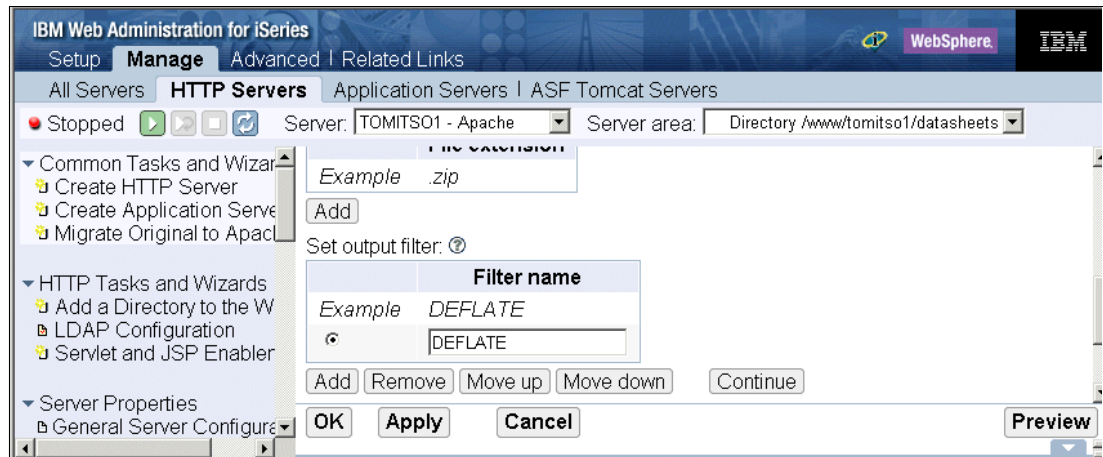


Figure 10-12 Compression: Output Filters tab Set output filter section

7. So far, you have configured the directory context to compress all files that are sent to the client. In the next steps, you will configure the equivalent directives for the BrowserMatch directives. These directives restrict compression for certain browser types.
- In the left navigation pane, under Server Properties, click **Request Processing**.

8. In the Request Processing panel (Figure 10-12), click the **Custom Environment Variables** tab.

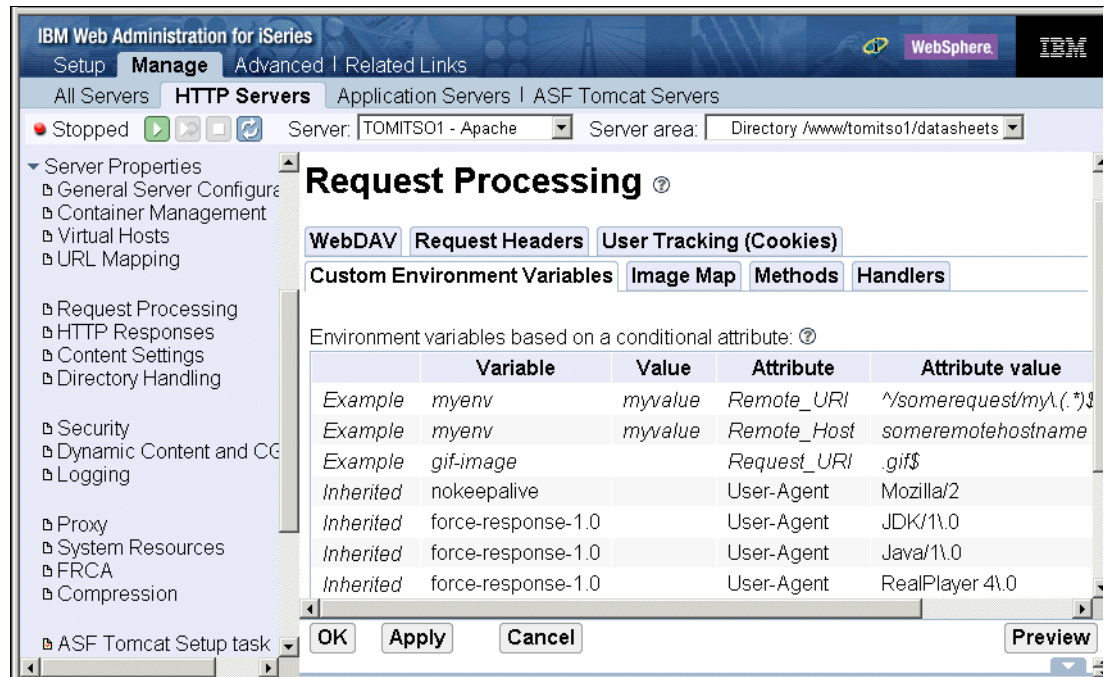


Figure 10-13 Request Processing: Custom Environment Variables page

9. On the Custom Environment Variables page, complete these steps:
 - a. Scroll down to the end of the table and click **Add** to add a new entry to the table.
 - b. Enter the following values into the input fields as shown in Figure 10-14:
 - Variable: gzip-only-text/html
 - Value: Leave empty
 - Attribute: User-Agent
 - Attribute value: ^Mozilla/4
 - Case sensitive: Select this option

Depending on the Case sensitive checkbox setting, the GUI creates a SetEnvIf or SetEnvIfNoCase directive.

These values generate the following directive:

```
SetEnvIf User-Agent ^Mozilla/4 gzip-only-text/html
```

The directive corresponds to the following BrowserMatch directive:

```
BrowserMatch ^Mozilla/4 gzip-only-text/html
```

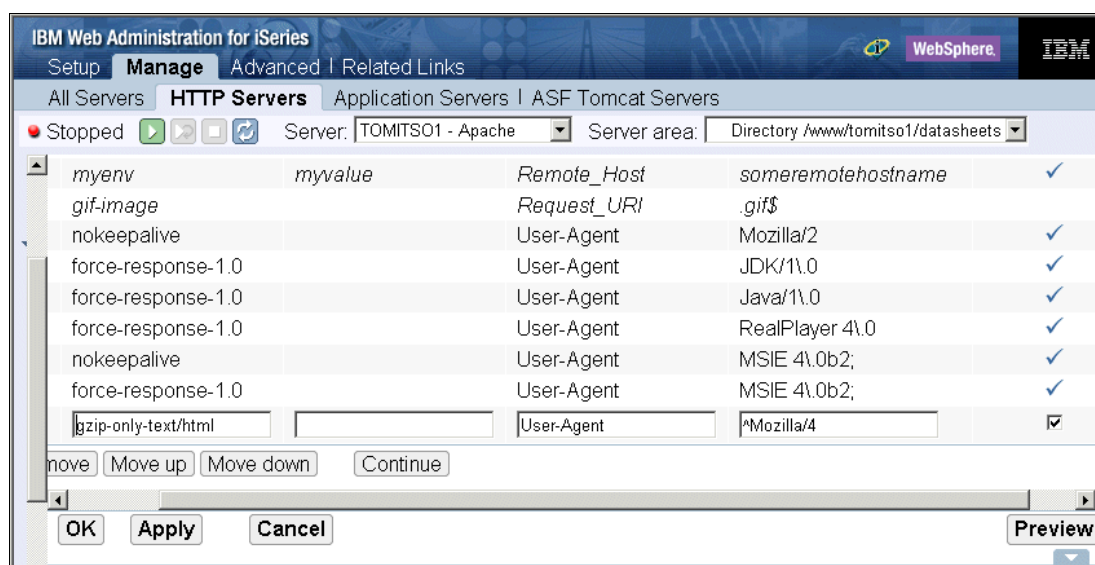


Figure 10-14 Request Processing: New entry on the Custom Environment Variables page

- c. Click **Continue**.
- d. Click **Add** to add another entry to the environment variables section.
- e. Enter the following values for the new entry.
 - Variable: no-gzip
 - Value: Leave empty
 - Attribute: User-Agent
 - Attribute value: ^Mozilla/4\0[678]
 - Case sensitive: Select this option

This entry further restricts compression in a way that compression is applied to Mozilla (Netscape) browser Version 4.06 to 4.08. The directive generated by the GUI is:

```
SetEnvIf User-Agent ^Mozilla/4\0[678] no-gzip
```
- f. Click **Continue** to save the entry.

g. Repeat the previous steps to add the following two entries:

- Variable: !no-gzip
- Value: Leave empty
- Attribute: User-Agent
- Attribute value: \bMSIE
- Case sensitive: Select this option

This entry allows compression for Microsoft Internet Explorer browsers. In fact, it removes the restriction that was previously introduced with the no-gzip option for all browsers that report themselves as Mozilla browsers. Also Internet Explorer reports itself as a Mozilla browser, but adds another value (MSIE) to the user agent string. The User-Agent variable is selected with the regular expression \bMSIE if the variable contains somewhere the value MSIE. If it does, compression is allowed via the !no-gzip parameter.

The second entry removes the restriction for text/html files.

- Variable: !gzip-only-text/html
- Value: Leave empty
- Attribute: User-Agent
- Attribute value: \bMSIE
- Case sensitive: Select this option

The directives generated by the GUI are:

```
SetEnvIf User-Agent \bMSIE !no-gzip
SetEnvIf User-Agent \bMSIE !gzip-only-text/html
```

h. Using the following parameter values, add the last two SetEnvIf directives to the HTTP server configuration:

- Variable: no-gzip
- Value: Leave empty
- Attribute: Request_URI
- Attribute value: \.(gif|jpe?g|png)\$
- Case sensitive: Deselect this option

The second directive is required to set the dont-vary variable.

- Variable: dont-vary
- Value: Leave empty
- Attribute: Request_URI
- Attribute value: \.(gif|jpe?g|png)\$
- Case sensitive: Deselect this option

The directives generated by the GUI are:

```
SetEnvIfNoCase Request_URI \.(gif|jpe?g|png)$ no-gzip
SetEnvIfNoCase Request_URI \.(gif|jpe?g|png)$ dont-vary
```


After all SetEnvIf and SetEnvIfNoCase directives are defined, the environment variables section should look like the example in Figure 10-15.

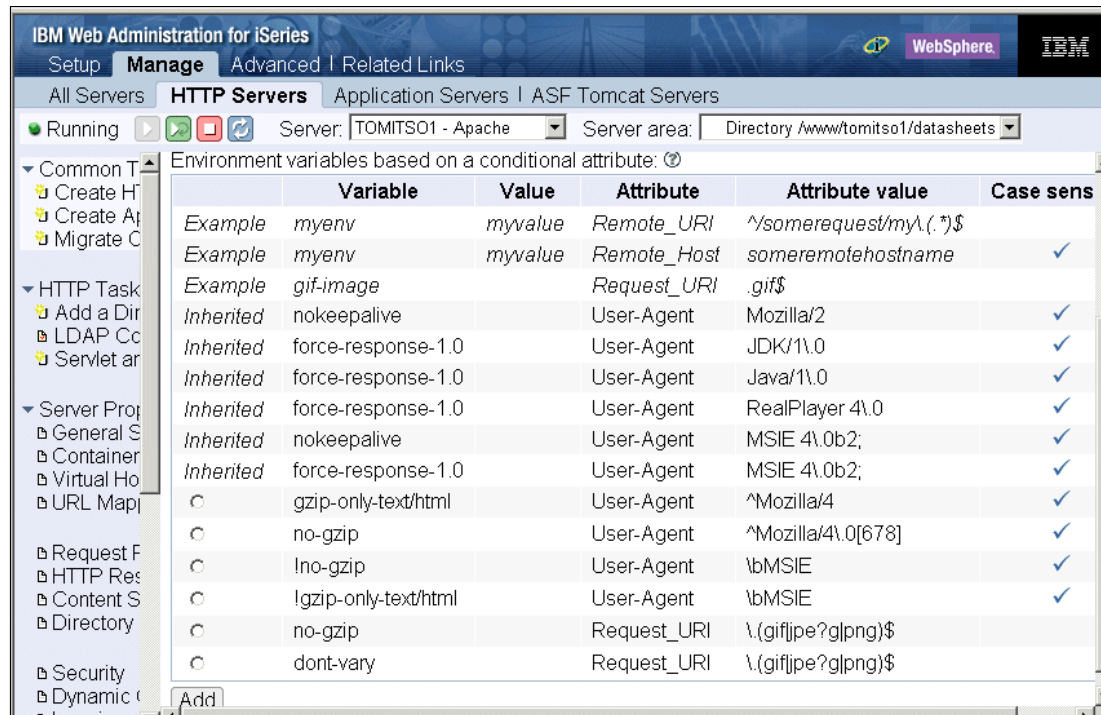


Figure 10-15 Request Routing: Custom Environment Variables

- i. Click **OK** to close the Request Processing window.

Note: When BrowserMatch directives already exist and you change custom environment variables via the IBM Web Administration for iSeries interface, the BrowserMatch directives are automatically converted into SetEnvIf directives.

10. The last configuration step prevents proxies to incorrectly handle the content. In the left navigation pane, click **HTTP Responses**.
11. In the HTTP Responses panel, click the **Response Headers** tab.

12. On the Response Headers page (Figure 10-16), follow these steps:

- a. Click **Add** to add a new entry to the Response headers section.
- b. Enter the following information:
 - Action: **Append**
 - Header name: Vary
 - Value: User-Agent
 - Environment variable: !dont-vary
- c. Click **Continue**.
- d. Click **OK** to save your settings.

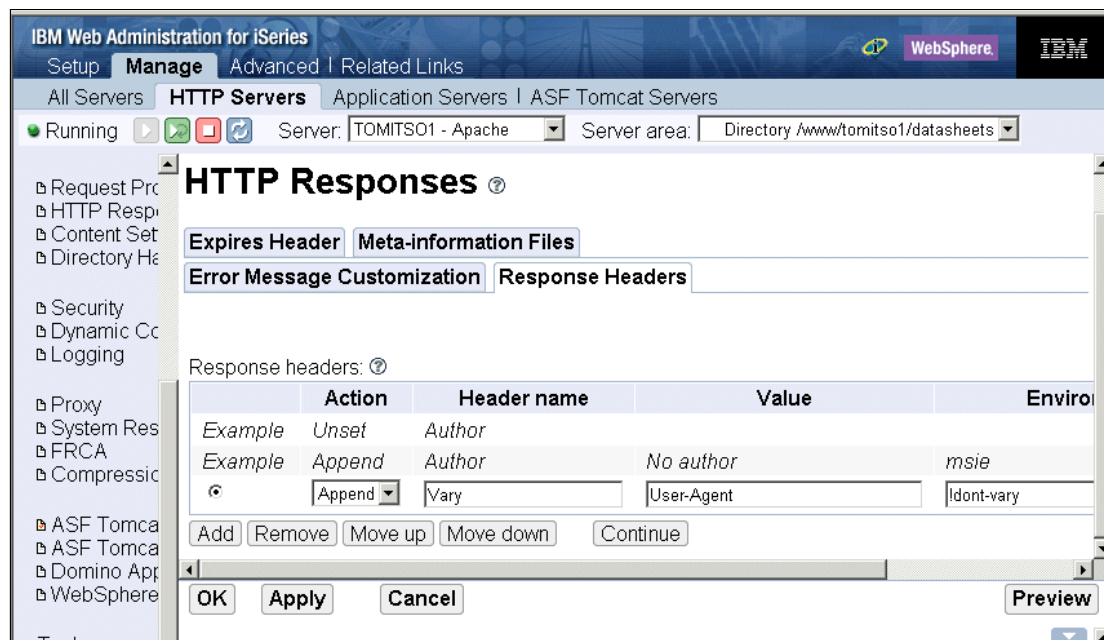


Figure 10-16 HTTP Responses panel

13. Restart the server to activate the new configuration.

Compressing only a few MIME types in a context

Using `AddOutputFilterByType` seems to be the preferred way to handle the idiosyncrasies of the many different Web browsers on the Internet. You can use a more complex example as demonstrated earlier. However, complexity invites the opportunity for error, either as part of your configuration or from the Web client's ability (or lack of ability) to handle a certain MIME type. For example, you may want to test to see which versions of Netscape can handle compressed PDFs over an SSL-encrypted session.

In this configuration scenario, only content of the following MIME types are compressed:

- ▶ text/html
- ▶ text/plain
- ▶ text/xml

Perform the following steps to activate outbound compression for these MIME types.

1. From the IBM Web Administration for iSeries interface, click the **Manage** tab.
2. From the Server list, select the server you want to configure.
3. From the Server area list, select the context you for which want to configure compression.

4. In the left pane under Server Properties, click **Compression**.
5. In the Compression panel, select the **Output Filters** tab.
6. On the Output Filters page (Figure 10-17), complete these steps:
 - a. Scroll down to the Add output filter by MIME type section and click **Add** to add a new entry.
 - b. Enter the following information to enable compression for the text/html MIME type:
 - MIME type: text/html
 - Filter name: DEFLATE
 - c. Click **Continue**.

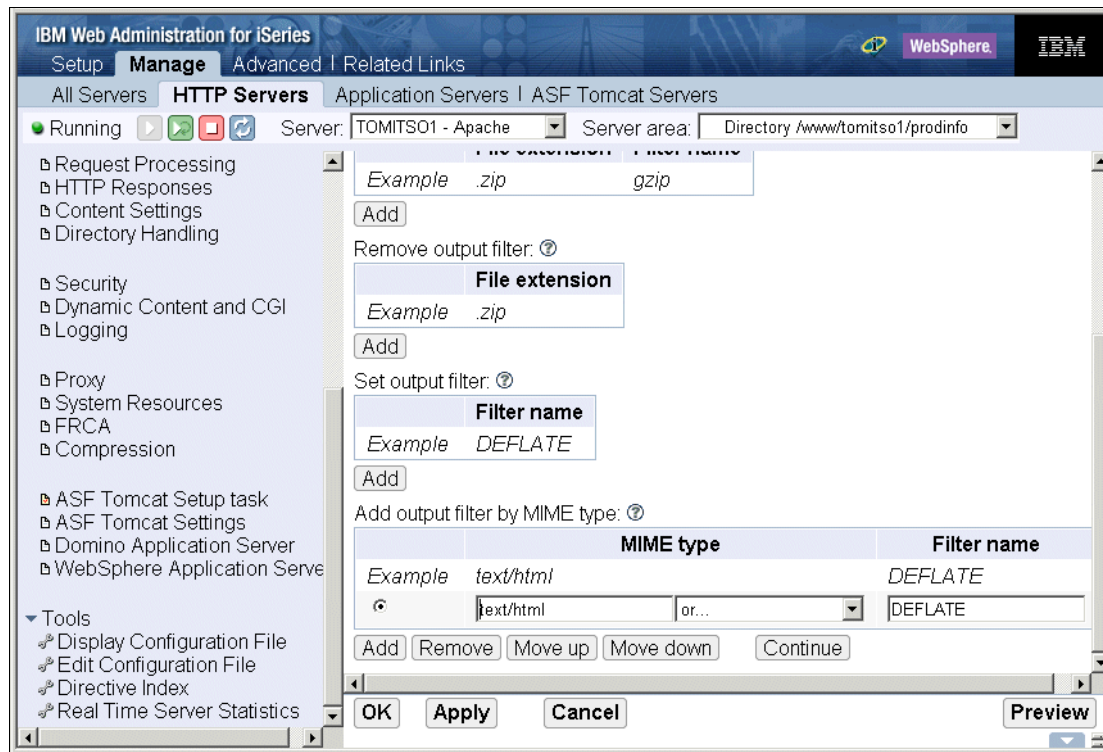


Figure 10-17 Compression: Adding an output filter by MIME type on the Output Filters page

- d. Click **Add** again to add the entry for the MIME type text/plain:
 - MIME type: text/plain
 - Filter name: DEFLATE
 - e. Click **Continue**.
 - f. Click **Add** again to add the entry for the MIME type text/xml:
 - MIME type: text/xml
 - Filter name: DEFLATE
 - g. Click **Continue**.
 - h. Click **OK**.
7. Restart the server to activate the new configuration.

Example 10-3 shows the key directives needed to support MIME type-based compression using the mod_deflate module in the httpd.conf file in bold. These directives were added by the IBM Web Administration for iSeries interface. Most of the other directives that do not directly affect this example were removed.

Example 10-3 mod_deflate: Minimal configuration directives

```
# Configuration originally created by Create HTTP Server wizard on Wed Sep 22 15:54:41 CEST
2004
2   LoadModule deflate_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
3   Listen *:8001
4   DocumentRoot /www/tomitsol1/htdocs
...
24  <Directory /www/tomitsol1/prodinfo>
25      Order Allow,Deny
26      Allow From all
27      AddOutputFilterByType DEFLATE text/html
28      AddOutputFilterByType DEFLATE text/plain
29      AddOutputFilterByType DEFLATE text/xml
30  </Directory>
```

This example uses mod_deflate to compress all the files served from the directory /www/tomitsol1/prodinfo (and all subdirectories) of the MIME types listed.

10.4.3 Logging

You can see information about how mod_deflate is performing by using:

- ▶ Custom deflate log file
- ▶ Communications trace
- ▶ HTTP server trace

Custom deflate log file

You can create a special deflate log file to capture the effectiveness of mod_deflate. This first configuration example simply reports the ratio of the output stream versus the input stream. Perform the following steps to define the custom log for compression.

1. From the IBM Web Administration for iSeries interface, click the **Manage** tab.
2. From the Server list, select the server you want to configure.
3. From the Server area list, select the **Global configuration** context.
4. In the left pane under Server Properties, click **Compression**.
5. In the Compression panel, select the **Advanced** tab.
6. On the Advanced page (Figure 10-18), complete these steps:
 - a. In the Deflate filter note section select the following:

- Filter note type: **Ratio**
- Filter note name: compratio

The specified parameter generates the following directive:

```
DeflateFilterNote Ratio compratio
```

The directive on its own does not cause any log entries to be written. Additional log options need to be configured to create the desired log.

- b. Click **OK** to save the configuration.

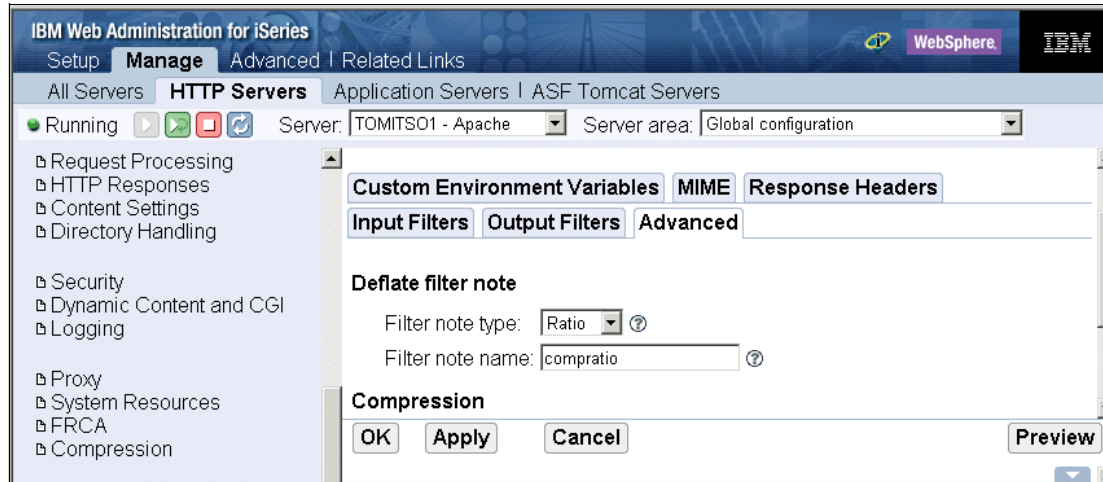


Figure 10-18 Compression: Advanced page Deflate filter note

7. In left navigation pane, click **Logging**.
8. In the Logging panel, click the **Custom Formats** tab.
9. On the Custom Formats page (Figure 10-19), complete these tasks:
 - a. Under Log formats, click **Add** to add a new log format entry.
 - b. Enter the following information:
 - Format name: deflate
 - Log format: %r %b (%{compratio}n)
 - c. Click **Continue**.
 - d. Click **Apply** to save the new entry.

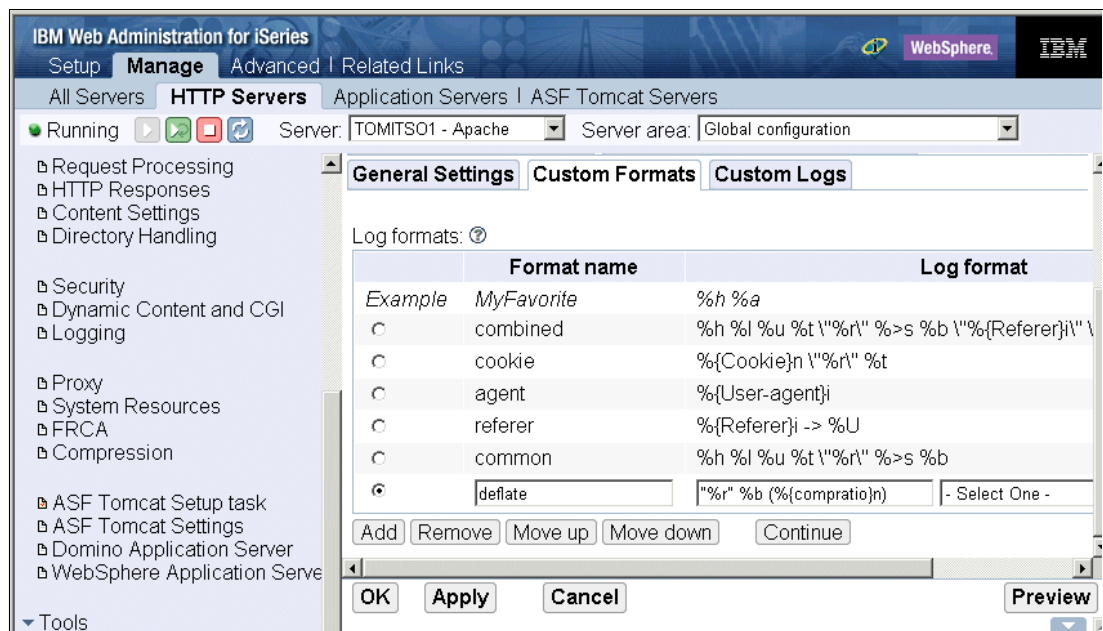


Figure 10-19 Logging: Custom Formats page

10. Click the **Custom Logs** tab.

11. On the Custom Logs page (Figure 10-20), complete these steps:

- a. Under Custom logs, click **Add**.
- b. Enter the following information:
 - Log: logs/deflate_log
 - Attributes / Log format: deflate
- c. You can also define additional attributes for log maintenance.

The Log format attribute relates to the custom format name that you defined in Figure 10-19.

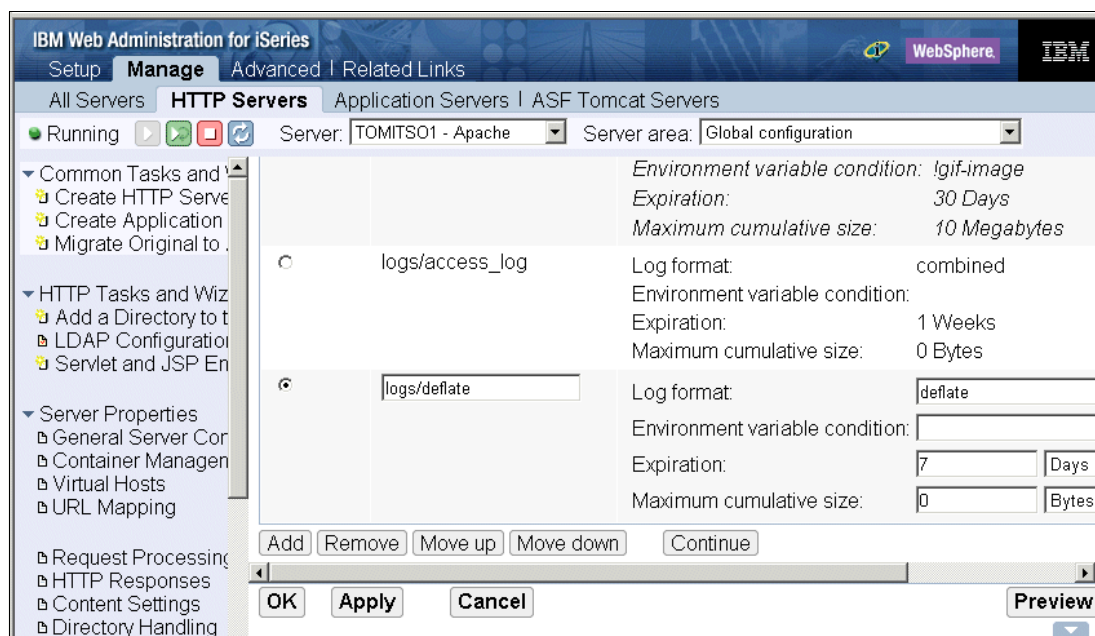


Figure 10-20 Logging: Custom Logs page

- d. Click **Continue**.
- e. Click **OK** to save the configuration.

12. Restart the server, access pages that are configured to be compressed, and open the deflate_log file to see how the log format looks like.

Example 10-4 shows the directives that were added to your Global configuration server context.

Example 10-4 mod_deflate: Simple deflate_log to report ratio of output stream to input stream

```
DeflateFilterNote Ratio compratio
LogFormat "%r %b (%{compratio}n)" deflate
CustomLog logs/deflate_log deflate
```

A portion of the deflate_log is shown in Example 10-5. After HTTP/1.1 are two numbers. The first is the number of bytes sent for this request. The second in parentheses is the ratio, which you can think of as a percentage.

The first line reads 2020 (18). This was for the file index.html, which in our application was originally 10 876 bytes. That is, mod_deflate caused that file to compress to a file that is approximately 18% of its original size.

For another example, consider the third line that reads 337 (101). This GIF file's original size is 314 bytes. `mod_deflate`, in this case, actually expanded the size of the file.

Another good example is the last line that reads 21057 (96). This JPEG file's original size is 21 705 bytes. `mod_deflate` in this case barely reduced the size of this JPEG file.

Example 10-5 mod_deflate: Simple deflate_log

```
"GET / HTTP/1.1" 2020 (18)
"GET /clearpixel.gif HTTP/1.1" 52 (79)
"GET /Background.gif HTTP/1.1" 337 (101)
"GET /Home_Hp3.gif HTTP/1.1" 337 (17)
"GET /Projects_Np1.gif HTTP/1.1" 260 (13)
"GET /People_Np1.gif HTTP/1.1" 252 (13)
"GET /Services_Np1.gif HTTP/1.1" 259 (13)
"GET /Products_Np1.gif HTTP/1.1" 264 (14)
"GET /SiteMap_Np1.gif HTTP/1.1" 270 (14)
"GET /Downloads_Np1.gif HTTP/1.1" 281 (14)
"GET /BuiltByNOF.gif HTTP/1.1" 1130 (67)
"GET /Home_NBanner.GIF HTTP/1.1" 671 (31)
"GET /Ss02043.JPG HTTP/1.1" 21057 (96)
```

Example 10-6 shows another more complex format that provides more information and more accurate results.

Example 10-6 mod_deflate: Accurate deflate_log to report ratio of output stream to input stream

```
DeflateFilterNote Input instream
DeflateFilterNote Output outstream
DeflateFilterNote Ratio compratio
LogFormat "%r %{outstream}n/%{instream}n (%{compratio}n)" deflate
CustomLog logs/deflate_log deflate
```

Restriction: The `LogFormat` and `CustomLog` directives can be defined through the IBM Web Administration for iSeries interface as explained earlier. However, the Advanced tab of the Compression page does not allow you to add more than one `DeflateFilterNote` directive. Therefore, if you want to set up the more complex compression logging scenario as shown in Figure 10-6, you have to edit the configuration file and enter the `DeflateFilterNote` directives manually.

As shown in Example 10-7, the `deflate_log` now produces more information. Three numbers are present after the HTTP/1.1 text. The first number represents the number of bytes output by `mod_deflate`. The second is the number of bytes as input to `mod_deflate`. And the third in parenthesis is the ratio.

Example 10-7 mod_deflate: More accurate deflate_log

```
"GET /People/people.html HTTP/1.1" 2069/15190 (13)
"GET /Background.gif HTTP/1.1" -/- (-)
"GET /clearpixel.gif HTTP/1.1" -/- (-)
"GET /Home_Np1.gif HTTP/1.1" -/- (-)
"GET /Products_Np1.gif HTTP/1.1" -/- (-)
"GET /Projects_Np1.gif HTTP/1.1" -/- (-)
"GET /SiteMap_Np1.gif HTTP/1.1" -/- (-)
"GET /Downloads_Np1.gif HTTP/1.1" -/- (-)
"GET /Services_Np1.gif HTTP/1.1" -/- (-)
"GET /a_SatelliteDataIcon_4.gif HTTP/1.1" -/- (-)
"GET /People_Hp3.gif HTTP/1.1" 331/1820 (18)
"GET /BuiltByNOF.gif HTTP/1.1" -/- (-)
"GET /People_NBanner.GIF HTTP/1.1" 727/2159 (33)
```

```
"GET /DataIcon.GIF HTTP/1.1" 162/175 (92)
"GET /People/a_ArrowLine.gif HTTP/1.1" 99/1620 (6)
"GET /Employees_Ns1.gif HTTP/1.1" 340/1821 (18)
```

Tip: In Example 10-7, some of the lines show “-/- (-)” for the effects of mod_deflate on the file being served. This is written to the log file when the if-modified-since rule sends a 304 (not modified) response.

Communications trace

Using the iSeries communications trace, you can see the HTTP headers that are affected by the use of the mod_deflate module. See 13.2.9, “Communications trace” on page 353, for information about how to start, stop, print, and then delete an iSeries communications trace.

Example 10-8 shows an example of the HTTP get request from a Web client to the HTTP Server (powered by Apache) that accepts compression for the file /index.html using mod_deflate. The HTTP header ACCEPT-ENCODING is highlighted in bold.

Example 10-8 HTTP get: Request showing ACCEPT-ENCODING of GZIP and DEFLATE

```
*E..A.*@.}..*.*.....*.*@+T***.***
*p.*.*.*.GET /INDEX.HTML HTTP/1.1*
*..ACCEPT: /*.*.REFERER: HTTP://A*
*$20:8000/SITEMAP/SITEMAP.HTML..A*
*$CCEPT-LANGUAGE: EN-US,ES-CO;Q=0.*
*5..ACCEPT-ENCODING: GZIP, DEFLAT*
*E..USER-AGENT: MOZILLA/4.0 (COMP*
*$ATIBLE; MSIE 6.0; WINDOWS NT 5.1*
*;*..NET CLR 1.1.4322)..HOST: AS20*
*:*8000..CONNECTION: KEEP-ALIVE...*
*..L.*
```

As shown in Example 10-9 here is the reply from the HTTP Server (powered by Apache). Some comments related to the headers that we have highlighted in bold.

- ▶ **CONTENT-ENCODING: GZIP:** This header tells the client how the information was encoded.
- ▶ **CONTENT-LENGTH: 2020:** This header is found on all replies. The interesting point is that the original /index.html file was 10867 bytes. This suggests a five-fold decrease in the size of the document.

Example 10-9 HTTP reply: Content headers indicating that the data was compressed

```
*E..*.*@.*.8.....*.*@.*.*+T***
*p.*.*.*.HTTP/1.1 200 OK..DATE: T*
*$UE, 24 JUN 2003 15:29:52 GMT..SE*
*$RVER: APACHE..LAST-MODIFIED: FRI*
*,$ 29 MAR 2002 01:56:13 GMT..ETAG*
*:* "1F601-2A73-35082940"..ACCEPT-*
*$RANGES: BYTES..VARY: ACCEPT-ENCO*
*$DING,USER-AGENT..CONTENT-ENCODIN*
*$G: GZIP..CONTENT-LENGTH: 2020..K*
*$EEP-ALIVE: TIMEOUT=300, MAX=95..*
*$CONNECTION: KEEP-ALIVE..CONTENT-*
*$TYPE: TEXT/HTML; CHARSET=WINDOWS*
*_-1252.....*.....*Z¢$*8..***.G*
*.*.*.*@HG.R*-*.*.*.*LVJ*K*.H***.*
*:*.*.*.*.*|..L:IOX.*.*.*.*$$.?U*.*V**
```



```
**.*R.*****.*p*0*.*!7***V**%*.2***
*0.*!**** **p**0**U**=**C**A* *****
*****<*..*AGHZ=*JJ*.*LZ*!*GD*T6*
...
```

Tip: A communications trace is a good tool for problem determination with mod_deflate. For example, if your Internet Explorer browser is configured for proxy, it may not be sending the accept-encoding HTTP header. The reason is most likely that your Web client is configured to use HTTP/1.0 for the proxy connection, which does not support deflate. A communications trace shows that the Web client is not sending in the accept-encoding HTTP header.

To configure your Internet Explorer Web client to use HTTP/1.1 for the proxy connections, select **Tools** → **Internet Options**. Click the **Advanced** tab and select **Use HTTP 1.1 through proxy connections**. If you select this options, you see the accept-encoding header in your communications trace.

HTTP server trace

You can turn on the HTTP server trace by using option -vv (verbose) to capture details about the processing of each file by mod_deflate. See 13.2.5, “HTTP server trace” on page 341.

By repetitively searching for Zlib in the spooled file created by the HTTP server trace, we found the statements shown in Figure 10-10. It is interesting to note that last line in Example 10-10 for URL /cgi-bin/MACRO1.MBR/run is the output of a Net.Data macro that was also compressed to about 25% of its original size.

Example 10-10 mod_deflate logging which files were compressed and to what size

```
000000ED:233144 Zlib: compressed 10867 to 2002 : URL /index.html.
000000EC:057400 Zlib: compressed 6751 to 1290 : URL /Products/products.html.
000000EB:114480 Zlib: compressed 15190 to 2069 : URL /People/people.html.
000000EB:265264 Zlib: compressed 10631 to 2869 : URL /cgi-bin/MACRO1.MBR/run.
```

10.4.4 Controlling the compression environment

In addition to the directives that control the kind of traffic that is going to be compressed or decompressed, you can configure directives that control the compression behavior of the mod_deflate module. The HTTP Server (powered by Apache) provides default values for these directives. Here is a brief description of the available directives:

- ▶ **DeflateBufferSize:** Specifies the size in bytes, kilobytes, megabytes, or gigabytes of the fragments that zlib should compress at one time. The default value is 8096 bytes.
- ▶ **DeflateCompressionLevel:** Specifies the level of compression to be used. The higher the value is, the greater the compression is. Higher compression levels require additional CPU time. The default level value specifies a level of compression that does not significantly increase CPU time on most systems. The default level value is 6.
- ▶ **DeflateMemLevel:** Specifies how much memory should be used by zlib for compression, in 16K increments. A value of 1 equates to 16K, while a value of 8 equates to 128K. The default value is 9 (144K).
- ▶ **DeflateWindowSize:** Specifies the zlib compression window size (the history buffer) in 16K increments. A value of 9 equates to 144K, while a value of 15 equates to 240K. Generally, the higher the window size is, the higher the compression ratio and greater usage of memory are. The default value is 15 (240K).

Note: Modifying these directives can have a positive or negative impact on performance and CPU load. The default values shipped with the system fit most environments. If you need to modify the compression environment, familiarize yourself with the compression module and the impact these directives have. You can find more information at:

- ▶ http://httpd.apache.org/docs-2.0/mod/mod_deflate.html
- ▶ http://www.gzip.org/zlib/zlib_docs.html

Using the IBM Web Administration for iSeries interface, you can modify these directives by performing the following steps:

1. Select your server and the context for which you want to configure compression.
2. From the left navigation pane, click **Compression**.
3. On the Compression panel, click the **Advanced** tab.
4. On the Advanced page (Figure 10-21), modify the performance related compression directives. Click **OK** to save the changes.

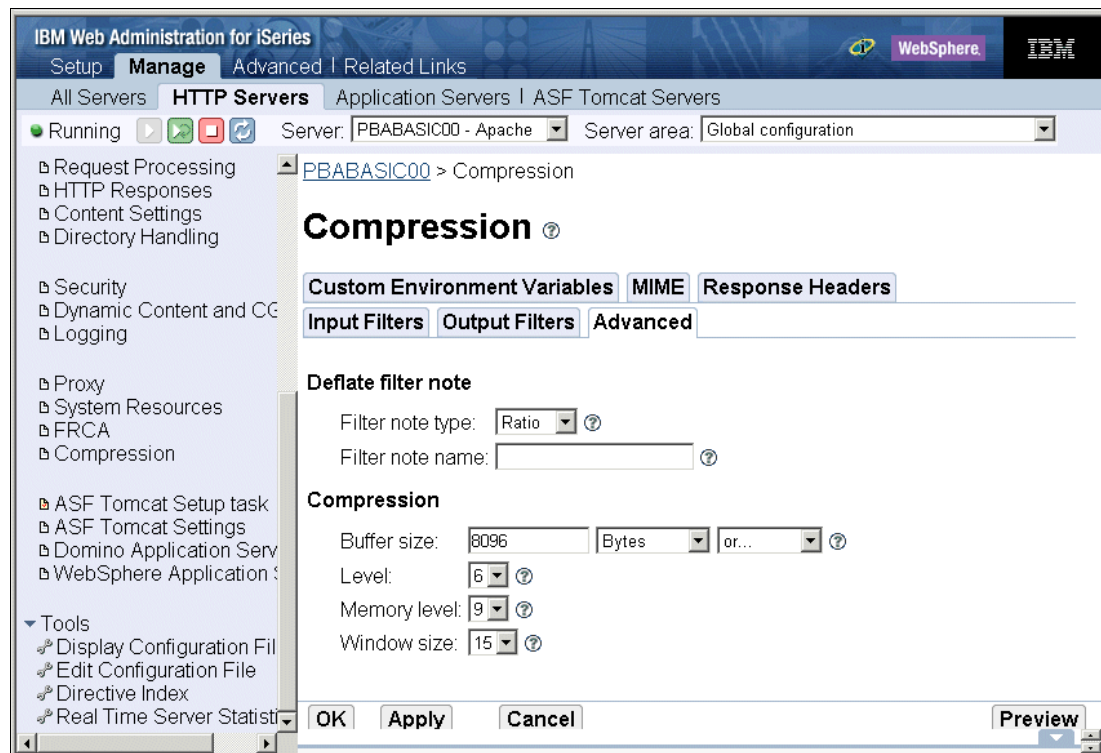


Figure 10-21 Compression: Advanced tab

5. Restart the server to activate the new settings.

10.4.5 For more information

There is documentation on the Apache Web site about mod_deflate that has information specific to setting up for compression. That site offers the best place to look for details:

http://httpd.apache.org/docs-2.0/mod/mod_deflate.html

The HTTP Documentation site has some documentation on the use of mod_deflate. See:

<http://www-1.ibm.com/servers/eserver/iseries/software/http/docs/doc.htm>

For more information about log formats, go to:

http://httpd.apache.org/docs-2.0/en/mod/mod_log_config.html#formats

10.5 Triggered Cache Manager

As demonstrated in Figure 10-3 on page 227, each time the client requests a page, the HTTP request must go through this process:

1. It goes through the network and passes each communication component such as firewall, routers, proxies, and so on.
2. It goes through the HTTP Server (powered by Apache) to identify that the request must be redirected to an application server.
3. The application server must in turn call the application.
4. The application processing the request may have to rely on many LOB database queries (or any other required tasks) to generate the dynamic content of the HTML page.
5. Even after the HTML is generated, this response must flow all the way back down the line.

If a different client or even the same client requests the information again, the whole process must be repeated. Even if the application has an application cache that can reduce the number of queries into the LOB database, the amount of processing required to get to the object stored in the application cache can be extremely large.

If you extend this environment to one with hundreds or even thousands of clients, CPU cycles (alone) required to process the requests increase considerably. At this point, it can be better if the implementation included a mechanism to serve client requests without going through the whole process every time.

In this implementation, create the dynamic content in a proactive fashion and copy it to the iSeries IFS or perhaps a router with a cache mechanism. Either way, now when the client sends the request, dynamic content is already cached in the iSeries IFS or router. In this way, the request can be processed with less demand on server resources and the response time for the client improved.

TCM is a component in the iSeries server that was created exactly for this purpose. This component is packaged in the IBM HTTP Server, 5722-DG1 Option 1. See Table 2-2 on page 20 for the details about how 5722-DG1 is packaged.

The TCM is a TCP server. It may be used in conjunction with Web servers and Web document caching agents to keep Web sites running at peak performance.

TCM is:

- ▶ A cache manager, not a cache or cache server
- ▶ Based on trigger messages

This means you must set up application triggers for the server to work. These messages are sent to the TCM via the HTTP/1.0 protocol.

- ▶ A stand-alone server that can work with multiple types of caching mechanisms, for example caching routers, proxy caches, and so on

It is useful in the environment we describe here for the HTTP Server (powered by Apache), but it can be used for the HTTP Server (original), Domino, or WebSphere Application Server environments as well.

TCM is proactive (based upon updates to an LOB database) in the update of Web content. TCM causes the Web content to be dynamically regenerated and then placed in a location (usually the iSeries IFS) that was configured to be served as static content by your Web server. The TCM is most effective for a Web site that has a large number of requests for content that is somewhat constant, but changes frequently.

One of IBM's first uses of the TCM concept was to drive the 1996 Summer Olympic Games Web site. Think of a downhill skiing results page (mostly HTML) that is composed of hundreds of dynamic items including names, times, scores, and so on for a particular event. If, for every request of that page, the application server had to re-run all the database queries to dynamically calculate the content. That level of activity had the potential to bog down the server with many repetitive actions. If only when the application LOB data was updated with new results, then the application server was used to update a standard results page to be served from a "static" portion of your Web site. This tremendously reduced the burden on the application server.

10.5.1 TCM system requirements

The TCM server requires the following components to run:

- ▶ TCP/IP Connectivity Utilities, 5722-TC1 licensed program
- ▶ IBM Developer Kit for Java, 5722-JV1 licensed program

Note: TCM does not require IBM Developer Kit for Java, 5722-JV1. It uses Java support shipped with OS/400. Currently the Start TCP/IP Server (STRTCPSVR) command erroneously checks for 5722-JV1. A V5R1 PTF (SI02889 for product 5722-SS1) corrects this problem. This level of support is built into OS/400 starting with V5R2.

- ▶ IBM HTTP Server, 5722-DG1 licensed program
- ▶ Triggered Cache Manager, 5722-DG1 option 1
- ▶ The latest 5722-DG1 group PTF:
 - V5R1: SF99156
 - V5R2: SF99098
 - V5R3: SF99099

Note: If you plan on using TCM with WebSphere Application Server (as the configured Data Source), then you need a special 5722-DG1 PTF:

- ▶ V5R1: SI09801
- ▶ V5R2: SI09799

This fix is integrated into i5/OS V5R3 of 5722-DG1.

The TCM server runs under its own user profile, which is QTCM.

The TCM configuration process should be done using the GUI. Do not edit the TCM configuration files directly. Configuration APIs are also available if you want to create your own configuration utility. See the following section.

10.5.2 TCM documentation

The following documents provide greater detail about the workings of TCM:

- ▶ A complete list of APIs to allow you to configure TCM in *HTTP Server for iSeries Programming*, GC41-5435. The configuration GUI that IBM provides for you as part of 5722-DG1 option 1 uses these APIs. Most likely, you do not have to use these APIs.

- ▶ Articles in the iSeries Information Center about TCM:

<http://publib.boulder.ibm.com/iseries/v5r3/ic2924/index.htm>

Then search for TCM.

- Triggered Cache Manager: An overview of TCM

Follow these links: **e-business and Web serving** → **HTTP Server** → **Concepts** → **Triggered Cache Manager**

- Trigger Messages: A description of the types and formats of the trigger messages that are sent to the TCM server. These messages use the HTTP/1.0 protocol.

Follow these links: **e-business and Web serving** → **HTTP Server** → **Concepts** → **Trigger messages**

- Set up Triggered Cache Manager

Follow these links: **e-business and Web serving** → **HTTP Server** → **Tasks** → **Triggered Cache Manager** → **Set up Triggered Cache Manager**

This article demonstrates how to use the default configuration that is shipped with TCM on the iSeries server. Depending on how your Web application is configured, this may be an easy way to see TCM working quickly.

This article also has a custom configuration example that leads you through all the options available when configuring TCM on the iSeries server.

Note: You can find a working example using a subset of all the complexity that is offered by a full implementation of TCM in 10.5.5, “Configuring a working TCM example” on page 264.

10.5.3 TCM directory structure and authorization

The TCM server has its own directory structure (Table 10-1), which is used to keep configuration and log files and various other data files required by the server.

Table 10-1 TCM directory structure

Directory	Description
/QIBM/UserData/TCM/instance/	This is the TCM root directory. Under this directory, the TCM server configuration process creates the directory used by each TCM server.
/QIBM/UserData/TCM/instance/ server_name	This is the root directory of your server_name TCM server.
/QIBM/UserData/TCM/instance/ server_name/daedalus.ini	This is the TCM configuration file.

The user profile QTCM requires *RWX data authorities and *ALL object authorities to the /QIBM/UserData/TCM directory and all files within it.

Tip: You should use only the configuration GUI or the APIs to alter the configuration files.

10.5.4 How the TCM server works

The steps involved before and during the TCM process are explained here:

1. Your HTTP server is configured to serve static content from a subdirectory in the IFS. The content in this subdirectory is undefined at this point. These files may be missing or contain old data.
2. Monitor for data changes.

Either your application or possibly a DB2® Universal Database™ (DB2 UDB) database trigger can be used to determine that data was changed. And, this is not just any data, but data upon which one or more dynamic Web pages depends.

Tip: It is beyond the scope of this IBM Redbook to explain how to write an application to create trigger messages. Here are some suggestions:

- ▶ Write a sockets client application that connects to the TCM server at the default port of 7049 and establish a connection. You have to simulate the HTTP/1.0 protocol, which is similar to the simulation we do by Telnet in this section. On the iSeries server, you can do this in any programming language.
- ▶ Write a Java application and take advantage of the `java.net.HttpURLConnection` and `java.net.URL` class files.

3. Send a trigger message to the TCM trigger handler with the information about the data that was changed.

Trigger request handlers perform the fundamental operations of a TCM server and are the key server feature. When a server starts, it creates a request handler for each description. The descriptions tell the server which type of trigger handler to create, the resources it must use, the process rules it must follow, and other various settings required for the particular type of defined handler. Trigger messages sent to a TCM server must address one of the request trigger handlers.

For example, TCM defines a standard trigger handler for administration named *admin*. As another example, if you configure a trigger handler by the name of `PRODUCTLIST`, the name created by TCM at startup is `TRH_PRODUCTLIST`.

TCM supports two types of trigger handlers:

- *Update Cache*: Tells the TCM server to create a trigger request handler that performs basic data transfers. It transfers (or copies) data from a data source to one or more cache targets. A description of this type contains:
 - A reference to a data source description, describing where data is located and how to obtain it.
 - References to one or more cache target descriptions, describing where the caches are located and how to work with them.
 - References to one or more acknowledgment target descriptions, describing where to send completion messages.
 - Various other settings required for trigger request handlers.

Tip: The easier choice is to use Update Cache. The data source is the Web application (WebSphere Application Server or Net.Data are two examples). The cache target is a subdirectory in the IFS.

- *Publish*: Tells the TCM server to create a trigger request handler that performs document publishing. A publish handler receives a list of document fragment names as input. It fetches each fragment from the configured data source, passes them through a dependency parser to update the object dependency graph, assembles the fragments into documents that can be served, and transfers (or copies) the fragments and documents to one or more cache targets. A description of this type contains everything an Update Cache type has as well as:

- A reference to an Object Dependency Graph (ODG) description, describing where persistent publishing data may be stored

TCM determines which pages need to be updated by consulting an ODG. This ODG is a data repository used to store dependency relationships for the pages the server handle. Therefore, when the data changes, TCM finds out what pages depend on the changed data. After all of the pages that are affected by the change are located, they are removed from cache and restored with the newly updated content. This then allows the dynamic update of the caching without restarting the server.

- A reference to a rule set description, describing how to process document fragments and servable documents and files
- Various other settings

4. Allow TCM to request the dynamic page from the data source.

When a TCM server needs to retrieve data (HTML documents, images, and video clips, for example) it uses information from a data source description to determine where the data is located and how to obtain it.

TCM supports two types of data sources:

- *File System*: Tells the TCM server it must use a file system to retrieve data. The server retrieves data by reading files from the iSeries IFS. A description of this type contains a directory path serving as the root to data files.

Tip: This option allows you to retrieve a file that is physically in the same iSeries server as is the TCM server. Using tools such as Network File System (NFS) or QFileSvr.400, the file can be physically located anywhere in your TCP/IP network.

- *HTTP Server*: Tells the TCM server it must communicate with a Web server to retrieve data. The TCM server retrieves data by using HTTP to request files from a Web server. A description of this type contains information about the system hosting the Web server, the TCP port the server is using, the URL path for the data files, and other required settings.

Tip: The HTTP server can be located anywhere in your TCP/IP network since it is referenced using a fully qualified URL. This includes sending the request to an HTTP server running on the same iSeries server that is running your TCM server.

5. Allow TCM to update the dynamic page content stored in the IFS or other cache targets.

When a TCM server needs to manage data in a cache, it uses information from a cache target description to determine where the cache is located and how to work with it. A description may list one or more targets. Each listed target is managed in a similar manner.

TCM supports three types of cache targets:

- *HTTP Server*: Tells the TCM server that it must communicate with a Web server to manage cache data. The server manages cache data by using HTTP to get, put, and delete files from a Web server. A description of this type contains a list of systems hosting Web servers, the TCP port the servers are using, the URL path for the cache data files, and other required settings.

Tip: While TCM on the iSeries server provides the mechanism to get, put, and delete files to a remote (or local) Web server, a complete solution for this option must include a CGI application running on the target HTTP server. IBM does not provide any example CGI applications. You are the one who *must* write this CGI application to handle the POST data.

- *Router*: Tells the TCM server that it must communicate with a router to manage cache data. The server manages cache data by using special protocols, such as External Cache Communication Protocol (ECCP) or Web Cache Control Protocol (WCCP), to work with files in the router's Web document cache. A description of this type contains a list of routers hosting Web document caches, the TCP port the routers are using, and various other settings.

Tip: The only routers we know that support this are the IBM Model 2212 and 2216 network routers via the ECCP protocol.

- *File System*: Tells the TCM server that it must use a file system to manage cache data. The server manages cache data by reading, writing, and deleting files in iSeries IFS. A description of this type contains directory paths serving as roots to data files.

This is the easiest method because TCM simply writes file into the IFS on your iSeries server. This is also the most practical solution. In this case the IFS can be thought of as a cache target for TCM. Remember, that TCM is not a cache, but a cache manager.

Tip: This option allows you to place a file into the IFS on the same iSeries server that the TCM server is. Using such tools as NFS or QFileSvr.400, you can physically move the file to anywhere in your TCP/IP network.

There are multiple scenarios where you can use TCM. One for example can be an HTTP server with one TCM server. Another example is multiple HTTP servers with multiple TCM servers. Or there can be a combination between HTTP and TCM servers. All scenarios occur on one or many iSeries servers.

10.5.5 Configuring a working TCM example

This section documents a TCM server working in cooperation with an existing Web application. The steps that are involved are:

1. Define the environment.
2. Create and configure the HTTP Server (powered by Apache).
3. Create and configure the TCM server.
4. Test the TCM server.
5. Test interaction between TCM and HTTP Server (powered by Apache).

Defining the environment

For an example of the power of TCM, we provide a fairly simple Web application as shown in Figure 10-22 on page 266. An HTTP Server (powered by Apache) named PBATCM00 serves both:

- ▶ Primarily static content from the iSeries IFS under the DocumentRoot of /tcp52d00/TCM/ITSOco. Specifically, a file in <DocumentRoot>/Products/products.html is also configured to be served as static content even though TCM is dynamically updating the products.html file whenever it receives a specially coded trigger message.
- ▶ A single dynamic page that is generated by an SQL select query against a table in an LOB database file. The URL to evoke the Net.Data application is:

`http://as20:8700/cgi-bin/MACR01.MBR/run`

A TCM server named TCMserv00 waits for a trigger message via the HTTP/1.0 protocol. When it receives this trigger message, it updates the static IFS file products.html from the dynamic content found in the LOB database table.

The following steps explain how this works. Each step corresponds to the number in Figure 10-22.

1. Both the TCM and HTTP Server (powered by Apache) servers are configured and started. The content of <DocumentRoot>/Products/products.html is undefined at this stage. That is, a file may be either missing or have old data in it from the last time it was updated.
2. A trigger message is sent to the TCMserv00 TCM server's Trigger Handler PRODUCTLIST. This message was most likely generated by an update to an SQL table that caused the static HTML in the products.html file to become out-of-date. Some events may include:
 - A database trigger, coded as part of the LOB database, detects that a table is updated.
 - An application running on the iSeries directly updates the LOB database.
 - An event occurs, such as the HTTP Server (powered by Apache) server has started.

The protocol for the trigger message is HTTP Version 1.0. The syntax of the trigger message is:

`-update -from /MACR01.MBR/run -to /products.html`

Notice the following explanation:

- **-update:** Indicates that this trigger message is to update a cache target
- **-from:** Defines the data source
- **-to:** Defines the cache target

3. The TCM server behaves as an HTTP client and prepares an HTTP GET request for the URL:

`http://as20:8700/cgi-bin/MACR01.MBR/run`

This URL is generated based on the TCM server's configuration (see Figure 10-22 for all the configuration values used in this scenario) and the text /MACR01.MBR/run, which immediately followed the -from parameter. In this case, the URL is:

`http://<Host>:<TCP Port>/<Root Directory>/MACR01.MBR/run`

4. The HTTP Server (powered by Apache) PBATCM00 is configured to evoke a Net.Data macro that dynamically generates HTML and returns it to the client.

Tip: In general, the URL `http://as20:8700/cgi-bin/MACR01.MBR/run` is a hidden way into your Web application. Normally your (non-TCM) Web clients do not link this URL in any HTML page.

5. The HTML result of the Net.Data macro invocation is returned to the TCM server by the HTTP Server (powered by Apache).
6. The TCM server then writes the HTML results as a file to the iSeries IFS:


```
/tcp52d00/TCM/ITSOco/Products/products.html
```

The path and name of this file are generated based on the TCM server's configuration (see Figure 10-22) and the text `/products.html`, which immediately followed the `-to` parameter on the trigger message. In this case, the path and name are:

```
<Target Directory>/products.html
```
7. The next and all subsequent requests for the URL `http://as20:8700/Products/products.html` are served as static HTML content by the HTTP Server (powered by Apache) PBATCM00.

Tip: You may be inclined to also use the HTTP Server (powered by Apache) local cache to further cache the resultant HTML file. But, HTTP Server (powered by Apache) local cache with dynamic update simply invalidates the local cache entry (forcing the HTTP server to go to the IFS anyway) and does not recache the file. And, worse, the dynamic update option for the local cache causes the HTTP Server (powered by Apache) to always check to see if the file is updated. This causes extra I/O operations, slowing down your HTTP server. See "What to cache?" on page 237 for more details about the local cache options.

FRCA local cache *does* work in this environment. This is because FRCA local cache automatically updates the contents of the NFC when a static file in the IFS changes. See 10.6.2, "How FRCA local cache works" on page 283.

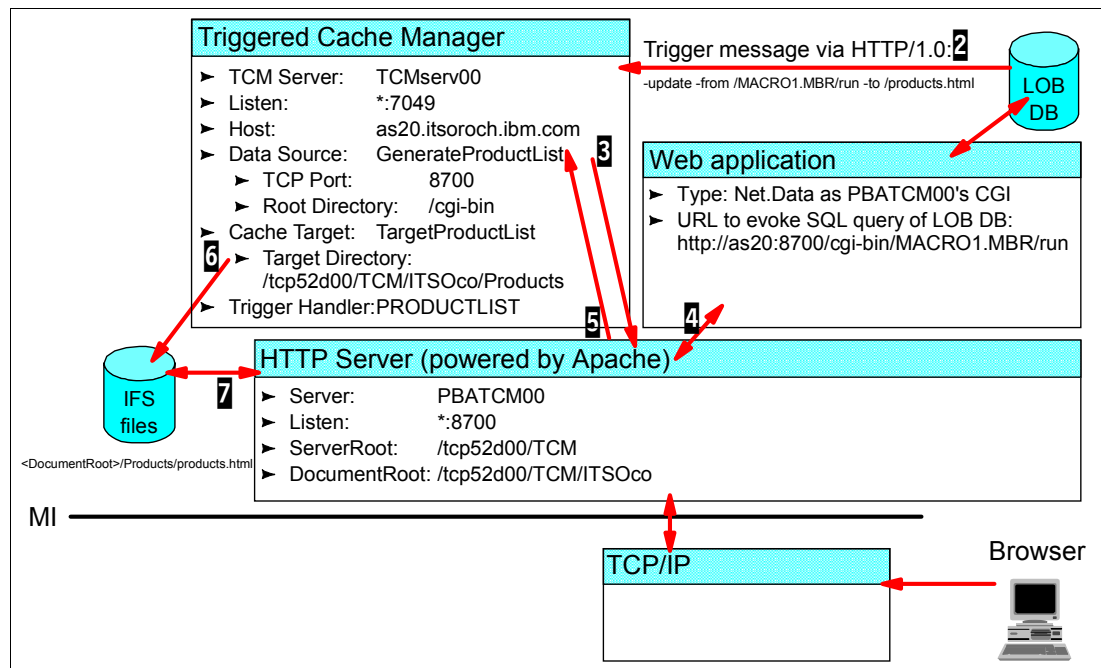


Figure 10-22 TCM: The environment defined

After you understand all the pieces of the configuration, the actual configuration steps are quite simple and straight forward.

Creating and configuring the HTTP Server (powered by Apache)

You must now create an HTTP Server (powered by Apache) with the characteristics defined in Table 10-2.

Table 10-2 TCM: HTTP Server (powered by Apache) used to serve Web application

Parameter	Value
Server name	PBATCM00
Server root	/tcp52d00/TCM
Document root	/tcp52d00/TCM/ITSOco
IP address	All
Port	8700
ScriptAlias	/cgi-bin/ /qsys.lib/tcp52lmast.lib/db2www.pgm/

Example 10-11 shows the final configuration file for the PBATCM00 HTTP Server (powered by Apache). This is a fairly standard configuration. Most of the work that this server does is to:

- ▶ Serve static files from the DocumentRoot /tcp52d00/tcm/itsoco and all subdirectories. One of the subdirectories that we are interested in is /tcp52d00/tcm/itsoco/products because this is where TCM will place the HTML results file named products.html.
- ▶ Using the ScriptAlias directive and the directives found within the <Directory /qsys.lib/tcp52lmast.lib/> content, allow this HTTP Server to evoke a Net.Data macro that does an SQL select into a table.

Example 10-11 TCM: Configuration file for PBATCM00 HTTP Server (powered by Apache)

```
# Configuration originally created by Apache Setup Wizard Tue May 27 21:30:43 UTC 2003
ScriptAlias /cgi-bin/ /qsys.lib/tcp52lmast.lib/db2www.pgm/
Listen *:8700
DocumentRoot /tcp52d00/tcm/itsoco
ServerRoot /tcp52d00/tcm
DefaultType text/plain
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes
-MultiViews
ErrorLog logs/error_log
LogLevel Warn
DirectoryIndex index.html
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log combined
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\0" force-response-1.0
SetEnvIf "User-Agent" "Java/1\0" force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\0" force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\0b2;" nokeepalive
SetEnvIf "User-Agent" "MSIE 4\0b2;" force-response-1.0
<Directory />
    Order Deny,Allow
    Deny From all
</Directory>
<Directory /tcp52d00/tcm/itsoco>
    Order Allow,Deny
    Allow From all
```

```
</Directory>
<Directory /qsys.lib/tcp52lmast.lib/>
  Order Allow,Deny
  Allow From all
  Options +ExecCGI
</Directory>
```

Creating and configuring the TCM server

Follow these steps to create a new TCM server named TCMserv00. After a new server is created, it may be custom configured and managed using the other forms available from the navigation frame.

1. From the IBM Web Administration for iSeries interface, click the **Advanced** tab and then the **TCM** subtab.
2. In the left pane, click **Create server**.
3. In the right panel, for Server Name, enter your TCM server name. In our example, this is TCMserv00. Leave the other parameters as the defaults. Click **Create**.

The screenshot displays the IBM Web Administration for iSeries interface. The top navigation bar includes 'Setup | Manage | Advanced | Related Links'. The 'Advanced' tab is selected, and the 'TCM' subtab is active. The left navigation pane shows 'Work with servers' and 'Create server' (selected). The main content area is titled 'IBM Triggered Cache Manager Server for iSeries' and contains a 'Create server' form. The form fields are: 'Server Name' (TCMserv00), 'Autostart' (radio buttons for Yes and No, with No selected), 'TCP Port (1-65535):' (7049), and 'Options' (radio buttons for 'Create with default configuration' and 'Create based on existing configuration', with the first option selected). Below the form are 'Create', 'Cancel', and 'Set to Defaults' buttons.

Figure 10-23 TCM: Creating the server TCMserv00

4. As shown in Figure 10-24, in the left pane, select your TCM server from the list. All the configuration settings that you specify are then applied to this TCM server so make sure that you select the correct ones.

Tip: You may have to refresh your Web client view of this frame to see your new TCM server. In Internet Explorer, press F5.

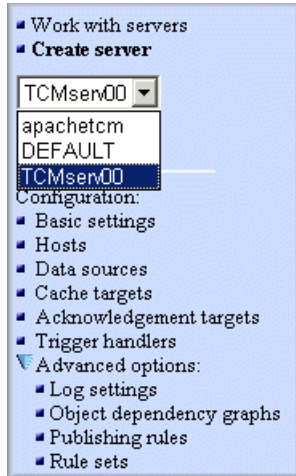


Figure 10-24 TCM: Selecting your TCM server by name

Now that the TCM server TCMserv00 is created, simply use the left pane to configure this server one object at a time. TCM allows for a flexible configuration with many named objects that can be used and reused by many different trigger monitors. Our path is simple.

5. As shown in Figure 10-25, click **Basic settings** to change any of the basic settings for the TCM server. For this example, we do not make any changes. At any point, you can click the help icon which is a little question mark (?) in a circle.

Tip: One of the interesting parameters is the number of threads. It specifies the number of concurrent threads that are spawned by the TCM server when it communicates with remote clients and servers. TCM operates as a multi-threaded server application. It is also queue-based because it uses messages between the threads to request work and manage work load. It is possible on a busy iSeries that a single or group of HTTP client requests for a dynamic page can take some time. As TCM waits for the HTML response from the Data Source, more work can queue up.

You can monitor these queues and the work involved to help you determine if changing this number to something bigger can help you for your Web application. See step b on page 277 for an example of determining the status of all the TCM queues.

The screenshot shows the 'IBM Web Administration for iSeries' interface. The top navigation bar includes 'Settings | Internet Users and Groups | Search Setup | TCM'. The left sidebar has a tree view with 'Work with servers' and 'Create server' expanded, showing 'TCMserv00' selected. Under 'Administration', 'Delete server' is listed. Under 'Configuration', 'Basic settings' is selected, along with 'Hosts', 'Data sources', 'Cache targets', 'Acknowledgement targets', 'Trigger handlers', and 'Advanced options'. The main content area is titled 'IBM Triggered Cache Manager Server for iSeries' and 'Change basic settings'. It shows the 'Server' as 'TCMserv00'. The 'Autostart' section has radio buttons for 'Yes' and 'No', with 'No' selected. The 'TCP Port (1-65535)' is set to '7049'. The 'Logging' section has radio buttons for 'Enabled' and 'Disabled', with 'Enabled' selected. The 'Additional Options' section includes 'Home Path' set to '/QIBM/UserData/TCM/instance/TCMserv00', 'Root Directory' set to 'root', and 'Retries' with radio buttons for 'No Maximum' (selected) and 'Maximum (0-999999)', with a text input field next to it. At the bottom are 'Change', 'Reset', and 'Set to Defaults' buttons.

Figure 10-25 TCM: Changing the basic settings

6. As shown in Figure 10-26, click **Hosts** to define configuration settings used to describe computer systems that host servers of interest to a TCM server. Computer systems (such as an iSeries server) may host a number of different server types. When a TCM server needs to communicate with another server, it reads a host description to obtain the IP address or host name of the system hosting the server. Again, use the help icon for more details.
7. In our case, we can use the default LOCALHOST since the host servers we will use are all on the same server. But, we create a host description for our local system. To do this, click **Create New Description** and enter your iSeries host name. Click **Create**.

Tip: You must list host names that can be resolved to IP addresses. That is, do not specify host names that do not exist.

The screenshot shows the 'IBM Web Administration for iSeries' interface. The top navigation bar includes 'Settings | Internet Users and Groups | Search Setup | TCM'. The left sidebar has a tree view with 'Work with servers' expanded, showing 'Create server' and a dropdown menu with 'TCMserv00'. Under 'Administration', 'Delete server' is listed. Under 'Configuration', 'Basic settings' is expanded, showing 'Hosts', 'Data sources', 'Cache targets', 'Acknowledgement targets', 'Trigger handlers', and 'Advanced options:'. The main content area is titled 'IBM Triggered Cache Manager Server for iSeries' and 'Create host description'. It shows 'Server: TCMserv00' and a text input field for 'Host Name' containing 'as20.itsoch.ibm.com'. At the bottom are buttons for 'Create', 'Cancel', and 'Set to Defaults'.

Figure 10-26 TCM: Creating a host description

8. Click **Data sources** to describe a data source to a TCM server. When the TCM server needs to retrieve data, it uses information from a data source description to determine where the data is located and how to obtain it.

TCM supports two types of data sources: file system and HTTP server. In our scenario, we use an HTTP Server as a data source.

9. Click **Create New Description**.

10. Complete the following steps (see Figure 10-27):

- a. In the Name field, type the name of your data source. In our case, we enter `GenerateProductList`.
- b. For the Type, select **HTTP Server**.
- c. Click **Next** to see the rest of the parameters needed for this data source.
- d. For Host, select **as20.itsoroch.ibm.com**, which was created earlier.
- e. For TCP Port (1-65535), type 8700, which is the port on which our HTTP Server (powered by Apache) Web application is listening.
- f. For the Root Directory, type `/cgi-bin`, which is a sort of prefix used to create the Uniform Resource Identifier (URI) for the data source. This can be `/`, but it requires every trigger message to be longer.
- g. Click **Create**.

The screenshot shows the 'IBM Web Administration for iSeries' interface. The main title is 'IBM Triggered Cache Manager Server for iSeries'. The sub-title is 'Create data source description'. The server selected is 'TCMserv00'. The form contains the following fields:

- Name:** GenerateProductList
- Type:** HTTP Server
- Host:** as20.itsoroch.ibm.com
- TCP Port (1-65535):** 8700
- Root Directory:** /cgi-bin

Below these fields is an 'Additional Options' section:

- Keep Alive:** ☐ Yes ☒ No
- Timeout (0-999999):** 0 seconds
- Threads (1-999999):** 1

At the bottom are three buttons: 'Create', 'Cancel', and 'Set to Defaults'.

Figure 10-27 TCM: Creating a data source description

11. Click **Cache targets** to configuration settings used to describe data caches to a TCM server. TCM supports three types of cache targets: HTTP server, router, and file system. In our scenario, we use the file system as the cache target.

12. Click **Create New Description**.

13. Complete the following steps (see Figure 10-28):
 - a. In the Name field, type the name of your cache target. In this case, we enter TargetProductList.
 - b. For Type, select **File System**.
 - c. Click **Next** to see the rest of the parameters needed for this cache target.
 - d. For Directory, type /tcp52d00/TCM/ITS0co/Products, which is sort of a prefix for the path in the IFS to place the result file. This can be “/”, but it requires every trigger message to be longer.
 - e. Click **Create**.



Figure 10-28 TCM: Creating a cache target description

14. In the left pane, Acknowledgement targets are descriptions of configuration settings used to describe where a TCM server sends completion messages after handling requests. When a TCM server completes a request, it may optionally send completion messages to inform someone (or something) that it handled a request. It uses information from an acknowledgment target description to determine where to send such messages. A description may list one or more targets. Each listed target is sent identical messages.

For this example, we do not specify an Acknowledgement target.

Click **Trigger handlers** to describe internal trigger request handlers for a TCM server. TCM supports two types of trigger handlers: publish and update cache. In our scenario, we use update cache.

15. Click **Create New Description**.

16. Complete the following steps (see Figure 10-29):
- In the Name field, type the name that you will use for your trigger handler. In our case, we enter **PRODUCTLIST**.
 - For Type, select **Update Cache**.
 - Click **Next** to see the rest of the parameters needed for this trigger handler.
 - For Data Source, select **GenerateProductList**.
 - For Cache Targets, select **TargetProductList**.

Tip: You can select multiple cache targets by holding down the Ctrl key and selecting each item in the list.

- For the remaining parameters, accept the defaults.
- Click **Create**.

IBM Web Administration for iSeries

Settings | Internet Users and Groups | Search Setup | TCM

Work with servers
Create server
TCMserv00
Administration:
Delete server
Configuration:
Basic settings
Hosts
Data sources
Cache targets
Acknowledgement targets
Trigger handlers
Advanced options:

IBM Triggered Cache Manager Server for iSeries

Create trigger handler description

Server: TCMserv00

Name: PRODUCTLIST
Type: Update Cache
Data Source: GenerateProductList
Cache Targets: << Select Cache Target >>
LOCAL_DIRECTORY
TargetProductList

Additional Options
Trigger Queue Collapse Policy: Collapse Identical Triggers

Create Cancel Set to Defaults

Figure 10-29 TCM: Creating a trigger handler description

This completes the configuration for the trigger handler **PRODUCTLIST**. The next step is to test the TCM server.

Testing the TCM server

First, you must start the TCM server and pass it a few simple requests to make sure it is running properly. These requests must be in the form of trigger messages.

Before continuing, we explain how TCM trigger messages are formed. First, TCM uses HTTP Version 1.0 (written as HTTP/1.0) as the protocol to carry the trigger messages. HTTP was chosen because it is a common protocol and easy to pass between systems and through firewalls. It is this requirement for HTTP/1.0 that leads us to the requirement that TCM's trigger messages be created by an application since there is really no native way on the iSeries server to generate the HTTP/1.0 protocol as a client. This application can reside on your iSeries or on any other system in your network.

Tip: It is beyond the scope of this IBM Redbook to explain how to write an application to create trigger messages. For this scenario, we use a Telnet VT100 client to simulate an HTTP client. You need a Telnet client that can locally echo the characters you are typing as you emulate an HTTP client when sending trigger messages to the TCM server. We used ZOC/Pro 4.11. You may download a 30-day evaluation copy from the Web at:

<http://www.emtec.com/main.html>

We made a configuration change to the defaults with ZOC. This was to change the session options to always "start session with local echo on". Select **Options** → **Edit Session Profile...** Select the **Device** tab. For the Telnet I/O Device, select **Start session with local echo on**. Click **Save**.

Second, we use the HTTP post method to send trigger messages to the TCM server. Here is an example for the syntax to query the /admin/ trigger handler for its -v[ersion].

```
post /admin/ http/1.0<Enter>
content-length: 3<Enter>
<Enter>
-v<Enter>
```

Tip: The syntax for the HTTP protocol is specific and unforgiving.

- ▶ <Enter> means to press the Enter key (or carriage return). And, yes, the third line in the sample above is asking you to press the Enter key all by itself.
- ▶ The content-length value must be precise. It should include all the characters including spaces and the <Enter> for the line that immediately follows it.
- ▶ If you make a mistake while typing the post syntax, the only way to fix it is start all over again. The Backspace key on your keyboard is not recognized by the TCM (nor any other HTTP) server.
- ▶ If you are going to use this Telnet client to emulate an HTTP client repeatedly, you may want to investigate using macros to record your keystrokes so you can accurately repeat them without making a mistake.
- ▶ The URI path information is case sensitive. For example, the following path causes the TCM server to send the HTTP/1.0 404 (file not found) to the client:

```
post /Admin/ http/1.0<Enter>
```

OK, now we are ready to start the TCM server and test it:

1. Start the TCM server. Click **Work with servers**, select the server you want to start (we selected TCMserv00) and click **Start** (see Figure 10-30). This process seems to take a bit of time. The server should *not* stay with a *Starting* status for more than one minute. Click **Refresh** until the status becomes *Active*.

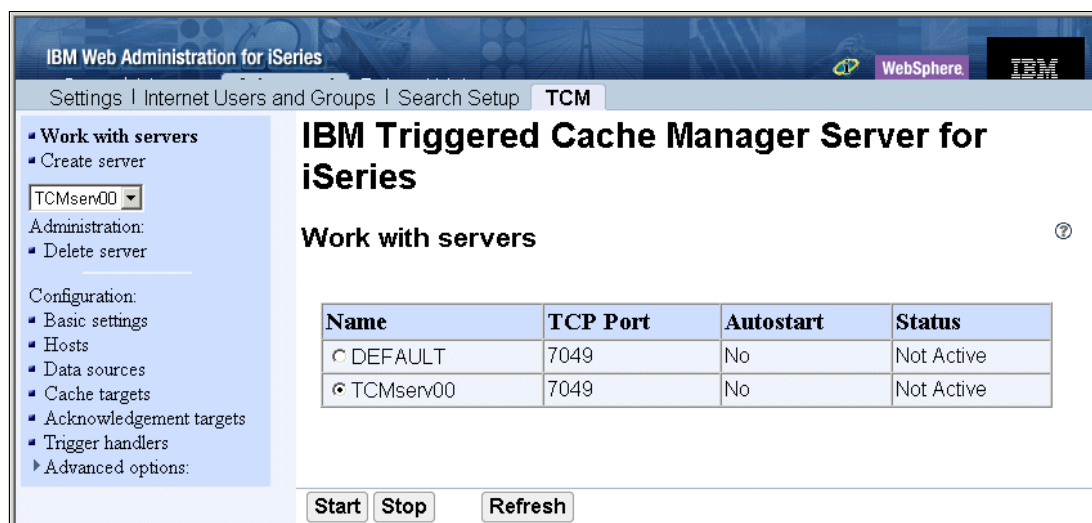


Figure 10-30 TCM: Working with servers and clicking Start

2. Determine the version of TCM that is running on your iSeries server. Follow these steps:
 - a. Use ZOC to Telnet to port 7049 on your iSeries server. Figure 10-31 shows the settings that we used for this client connection. Click **Options** to turn on local echo if it is not already enabled for this terminal session.

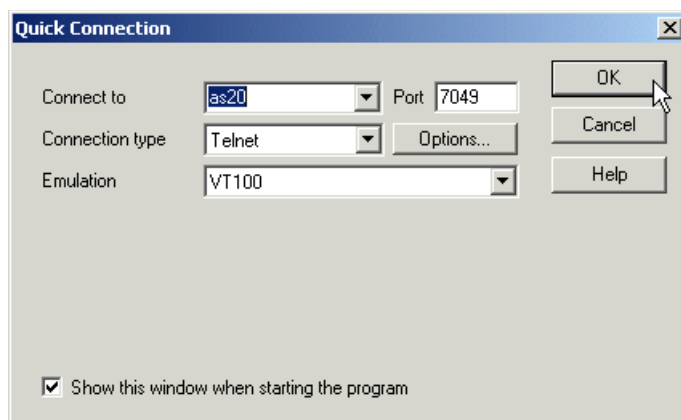


Figure 10-31 TCM: Using ZOC to Telnet to the iSeries TCM server at port 7049

- b. In ZOC, type the following HTTP/1.0 syntax. -v is a shorthand notation for -version.


```
post /admin/ http/1.0<Enter>
content-length: 3<Enter>
<Enter>
-v<Enter>
```

The TCM server should reply with something similar to:

```
HTTP/1.0 200
content-type: application/x-trigger-msglist
Content-length: 75
```

```
1152 0 0 admin ! Version: Daedalus 04042000A Started: 09/24/2004 14:44:37
[TELNET] INFO: DISCONNECTED
```

Tip: Daedalus was the IBM code name for the project that created TCM.

3. Determine the status of all the TCM queues. Follow these steps:

- a. Use ZOC to Telnet to port 7049 on your iSeries server.
- b. In ZOC, type the following HTTP/1.0 syntax:

```
post /admin/ http/1.0<Enter>
content-length: 8<Enter>
<Enter>
-queues<Enter>
```

The TCM server should reply with something similar to:

```
HTTP/1.0 200
content-type: application/x-trigger-msglist
Content-length: 1107

1140 1 1 admin ! SNK_LOCAL_DIRECTORY: active=0 queued=0 lifetime-total=0
lifetime-failed=0 lifetime-retried=0 threads=5
1140 1 1 admin ! SNK_TargetProductList: active=0 queued=0 lifetime-total=0
lifetime-failed=0 lifetime-retried=0 threads=5
1140 1 1 admin ! TRH_UPDATE_CACHE: active=0 queued=0 lifetime-total=0
lifetime-failed=0 lifetime-retried=0 threads=10
1140 1 1 admin ! SRC_LOCAL_HTTP: active=0 queued=0 lifetime-total=0
lifetime-failed=0 lifetime-retried=0 threads=5
1140 1 1 admin ! TRH_PUBLISH: active=0 queued=0 lifetime-total=0 lifetime-failed=0
lifetime-retried=0 threads=10
1140 1 1 admin ! admin: active=0 queued=0 lifetime-total=2 lifetime-failed=0
lifetime-retried=0 threads=0
1140 1 1 admin ! TRH_PRODUCTLIST: active=0 queued=0 lifetime-total=0
lifetime-failed=0 lifetime-retried=0 threads=10
1140 1 1 admin ! SRC_GenerateProductList: active=0 queued=0 lifetime-total=0
lifetime-failed=0 lifetime-retried=0 threads=5
1140 1 1 admin ! DeferQueue: active=0 queued=0 lifetime-total=0 lifetime-failed=0
lifetime-retried=0 threads=-1
1141 1 1 admin ! Lifetime total server requests=2
[TELNET] INFO: DISCONNECTED
```

As you can see, TCM uses many queues and many different handlers to operate even a simple configuration like ours. The one we are interested in is TRH_PRODUCTLIST, which is TCM's name for the trigger handler PRODUCTLIST we created in "Creating and configuring the TCM server" on page 268.

Testing interaction between TCM and HTTP Server (powered by Apache)

For this second part of the test, we start the HTTP Server (powered by Apache) and test the interaction. For this step, we send the trigger handler TRH_PRODUCTLIST a trigger message:

```
-update -from /MACR01.MBR/run -to /products.html
```

This trigger message causes TCM to make an HTTP request to the URL:

`http://as20:8700/cgi-bin/MACR01.MBR/run`

And then TCM places the resulting HTML in the iSeries IFS path and file. Refer to Figure 10-22 on page 266 to see how the configuration of TCM generates the desired output.

`/tcp52d00/TCM/ITS0co/Products/products.html`

Follow these steps to perform this task:

1. Start the HTTP Server (powered by Apache) PBATCM00.
2. Verify for yourself that the directory `/tcp52d00/TCM/ITS0co/Products/` is empty. If it is not empty, you may want to delete the file **Products.html** as a way to prove to yourself that TCM has dynamically generated the file. Choose one of the following options:

- Map a network drive to the iSeries IFS or use the Work with Object Links (WRKLNK) command as follows:

```
WRKLNK OBJ('/tcp52d00/TCM/ITS0co/Products/*')
```

If no objects are in the directory `/Products`, you should see the Work with Object Links display (Figure 10-32).

```
Work with Object Links

Directory . . . . : /tcp52d00/TCM/ITS0co/Products

Type options, press Enter.
  2=Edit  3=Copy  4=Remove  5=Display  7=Rename  8=Display attributes
  11=Change current directory ...

Opt  Object link          Type      Attribute  Text

(Cannot find object to match specified name.)
```

Figure 10-32 Work with Object Links (WRKLNK) of /Products subdirectory

- Use a Web client to request the `products.html` document directly as shown in Figure 10-33. Use the URL:

`http://as20:8700/products/products.html`

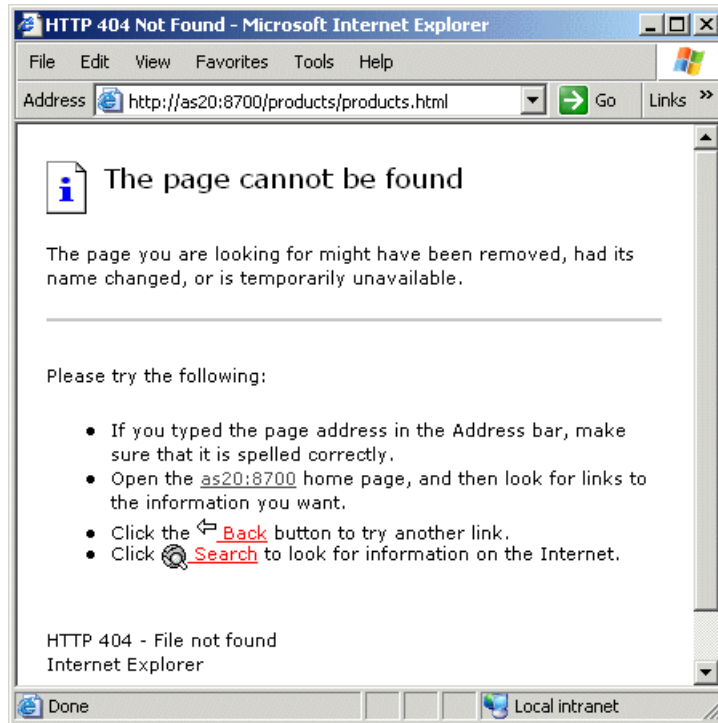


Figure 10-33 TCM: The `/Products/products.html` page is not found (yet!)

3. Send the trigger message to the TCM server. Follow these steps:

- a. Use ZOC to Telnet to port 7049 on your iSeries server.
- b. In ZOC, type the following HTTP/1.0 syntax:


```
post /TRH_PRODUCTLIST/ http/1.0<Enter>
content-length: 49<Enter>
<Enter>
-update -from /MACR01.MBR/run -to /products.html<Enter>
```

The TCM server should reply with something similar to:

```
HTTP/1.0 202
content-type: application/x-trigger-msglist
Content-length: 58

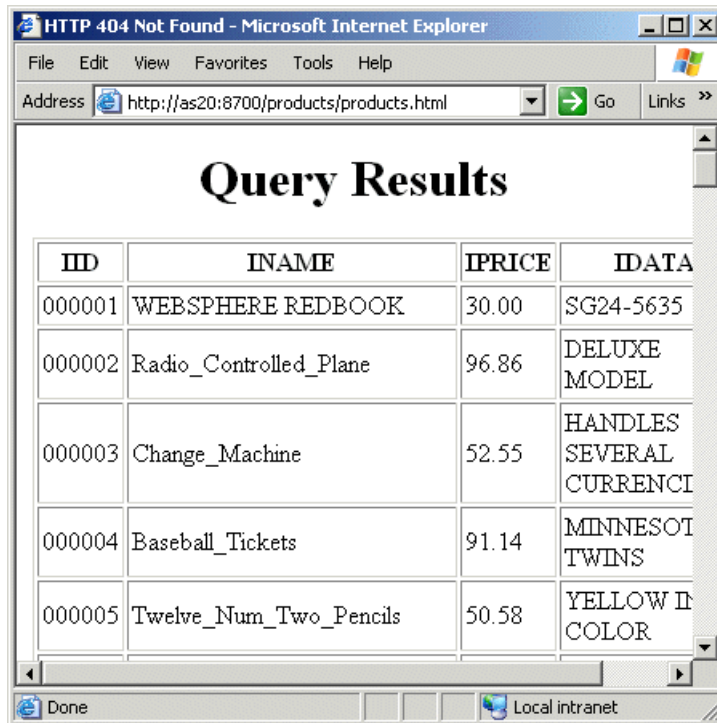
1102 2 2 TRH_PRODUCTLIST ! update-sink request is queued
[TELNET] INFO: DISCONNECTED
```

As you can see, TCM queues the request on the `TRH_PRODUCTLIST` trigger handler.

Tip: The URI path information on the post is case sensitive.

4. To verify that the directory /tcp52d00/TCM/ITSOco/Products/ now has a file Product.html, use a Web client as shown in Figure 10-34 using the URL:

<http://as20:8700/products/products.html>



IID	INAME	IPRICE	IDATA
000001	WEBSPHERE REDBOOK	30.00	SG24-5635
000002	Radio_Controlled_Plane	96.86	DELUXE MODEL
000003	Change_Machine	52.55	HANDLES SEVERAL CURRENCI
000004	Baseball_Tickets	91.14	MINNESOTA TWINS
000005	Twelve_Num_Two_Pencils	50.58	YELLOW IN COLOR

Figure 10-34 TCM: The /Products/products.html page created by TCM's actions

If this was a real Web application, the trigger message that was sent to the trigger handler TRH_PRODUCTLIST would be sent by your custom-written application. At this point, the HTTP Server (powered by Apache) PBATCM00 serves the static results file /Products/Products.html at static file speeds.

When the raw data in the SQL table is updated (for example, if the price of one of the items changes or a new item is added to the table), your application needs to send a new trigger message to the trigger handler to cause the /Products/Products.html file to be updated.

10.6 Fast Response Cache Accelerator

Note: With permission from iSeries Network, we include material from an article written for iSeries Network as a basis for this section. For the original article, see:

<http://www.iseriesnetwork.com>

FRCA (affectionately pronounced “Frica” by the Rochester developers) is a significant leap forward in caching architecture for your HTTP Server (powered by Apache). FRCA is dramatic in two ways. First, it can serve Web content at a whopping seven times faster than a file from the IFS. And, it serves over four times faster than the traditional local cache in the HTTP Server (powered by Apache). Second, FRCA does this with approximately four to seven times less CPU per transaction.

To back up this claim, see *iSeries Performance Capabilities Reference Version 5, Release 3*, SC41-0607, which is the center of release-to-release performance information about the iSeries server. You can find it on the Web at:

<http://www-1.ibm.com/servers/eserver/iseries/perfmgmt/resource.htm>

In this publication, see Section 6.1, which is dedicated to the HTTP Server (powered by Apache). Table 10-3 is a subset of the information provided.

Table 10-3 iSeries Web serving capacity planning various transaction types

Transaction type	Capacity Metric: trans/sec per CPW	CPU time Metric: CPW per trans/sec
Static Page (IFS)	1.75	0.57
Static Page (cache)	2.79	0.35
Static Page (FRCA)	13.01	0.08

For a rough guide as to the number of static pages you can serve from either the IFS, cached (from the local cache of the HTTP Server (powered by Apache)) or the new FRCA cache, multiply the published CPW of your iSeries server by the number you find in the Capacity column.

For example, if you take an average iSeries Model 810 with 1470 Commercial Processing Workloads (CPWs), you should expect the HTTP Server (powered by Apache) to serve around 2570 static pages from the IFS per second. For the same system, you should expect 4100 from the HTTP Server (powered by Apache) local cache. For FRCA, expect 19,100!

The CPU Time column is equally exciting. If, on the same Model 810, you need to serve 2000 static pages per second, you can expect 1140 CPWs consumed every second served from the IFS. The same files served from the local cache consume 700 CPWs per second. For FRCA, expect just 160 CPWs per second!

FRCA provides a *Fast Response Cache*, and it *accelerates* the overall performance of your system by freeing up your main processor or processors to do other things. To see where FRCA fits into the overall performance picture of your HTTP Server (powered by Apache), see Figure 10-3 on page 227.

FRCA, as we will see, can also cache “dynamic” content that you expire using a timer. But, while TCM seems like the better choice for caching dynamic content, it requires programming to make it work. FRCA, on the other hand, is just simple configuration.

10.6.1 What FRCA is

FRCA is the external product name given to a software architecture named Adaptive Fast Path Architecture (AFPA) developed by IBM Research. This architecture dramatically improves the capacity/performance of Web and other TCP servers.

Only one application on the iSeries server takes advantage of FRCA in OS/400 V5R2 and i5/OS V5R3. That is the HTTP Server (powered by Apache). That is, the architecture can be used by any iSeries TCP application, but only the HTTP Server (powered by Apache) has done so to date. To be clear, FRCA requires OS/400 at V5R2 or higher, plus the LPP IBM HTTP Server (5722-DG1).

FRCA directives are simply embedded within the HTTP Server (powered by Apache) configuration file (`httpd.conf`). This enables your HTTP server to use the FRCA cache. FRCA can be enabled for each listen port in the server configuration. This allows you to make a choice whether you use FRCA cache for each Listen on a specific <IP address:port>.

You can cache static content by specifying a specific file name or a group of files using wild cards, such as the asterisk (*). The loading of the cache occurs during HTTP server startup or at runtime depending on your configuration. Here is an example:

```
FRCACacheLocalFileStartup /ITS0/ITS0co/Downloads/*.html
```

Dynamic content, such as result of an HTTP request to a *content server*, can be cached by specifying a URI to identify the request that is then mapped to a fully qualified URL. This is a reverse proxy cache support. It allows you to access an HTTP server on this same iSeries or anywhere on your intranet or Internet to provide dynamic content that is automatically cached in the NFC. A timer is used to determine when cached items are stale. See the following example:

```
FRCAProxyPass /cgi-bin/ http://as21.domain.com:9999/cgi-bin/  
FRCAProxyCacheRefreshInterval /cgi-bin/ 180
```

At this point, it is important for you to realize that FRCA is two vastly different caching mechanisms wrapped into one package. That is, FRCA is both a:

- ▶ Local cache for IFS files that are generally static in nature.
- ▶ Reverse proxy cache for content that was generated by a dynamic content server either running on your local iSeries server or connected via a TCP/IP network.

Content server: A content server can be any content created by a Web application using the HTTP protocol. Good examples are:

- ▶ WebSphere Application Server serving servlets and Enterprise JavaBeans (EJBs)
- ▶ Tomcat serving servlets
- ▶ A CGI application, for example, Net.Data or Hypertext Preprocessor (PHP)

10.6.2 How FRCA local cache works

This section shows a series of diagrams that demonstrate the dramatic shift in processing from above the MI to below as we follow each client get request.

FRCA: Local cache miss scenario

Figure 10-35 shows a FRCA local cache miss scenario. The steps are:

1. An HTTP request is received by TCP and passed to FRCA.
2. FRCA intercepts the HTTP request and passes it to the SLIC HTTP Server code.
3. The SLIC HTTP server code parses the HTTP request and uses the URL as a search key into the hash table, one per server instance.
4. When the HTTP logical cache lookup fails, the HTTP request is redirected to the HTTP Server (powered by Apache) using the normal sockets interface.
5. The HTTP Server (powered by Apache) parses the HTTP request, maps the URL to an IFS file, builds the HTTP response from the IFS file, and calls Sockets send() to send the HTTP response. This is business as usual for the HTTP Server (powered by Apache).
6. After sending the HTTP response, the FrcaLoadFile() system API is called to load the file in the NFC.
7. FRCA calls the NFC to load a single copy of the file in the Network File Cache.

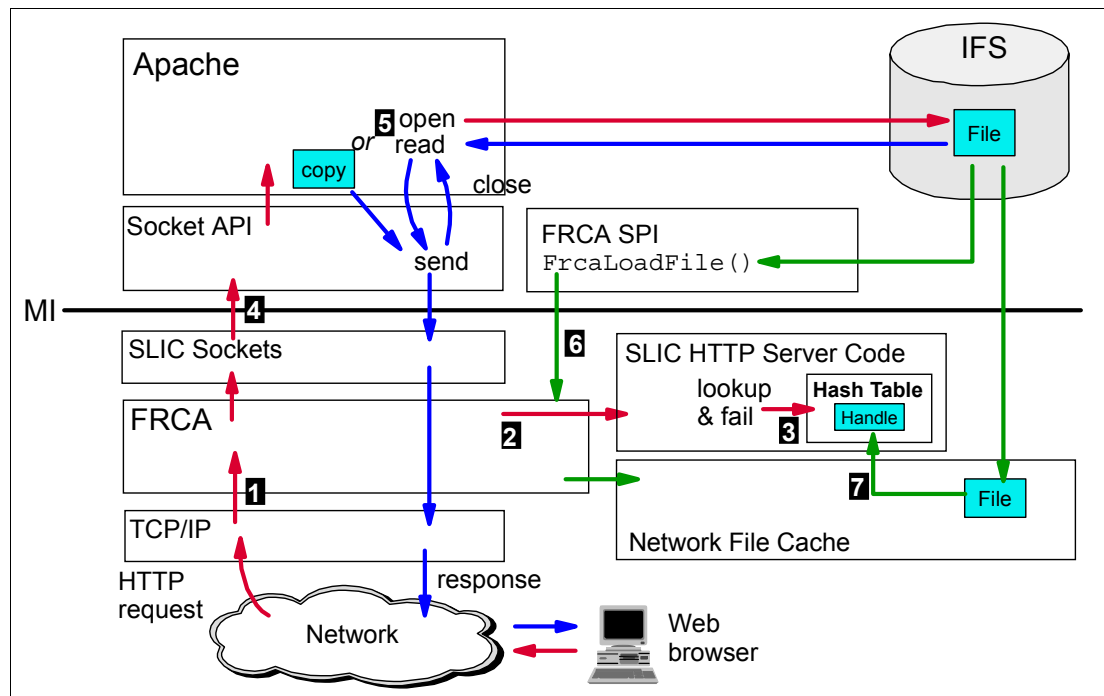


Figure 10-35 FRCA: Local cache miss scenario

FRCA: Local cache hit scenario

Figure 10-36 shows a FRCA local cache hit scenario. The steps from request through response are:

1. An HTTP request is received by TCP and passed to FRCA.
2. FRCA intercepts the HTTP request and passes it to the SLIC HTTP server code.
3. The SLIC HTTP server code parses the HTTP request and uses the URL as a search key into the hash table.
4. When the HTTP logical cache lookup is successful, NFC is called to locate the file data using the NFC handle found in the hash table.
5. NFC finds the file using the handle and returns it to the SLIC HTTP server code.
6. The SLIC HTTP Server code builds the HTTP response header, links the file data to it, and sends it as a response through TCP/IP.

Tip: FRCA local cache has an interesting feature. Assume that a file located in the IFS is cached in the NFC by FRCA. If that file is updated in the IFS, it is also automatically updated in the NFC and the new content is served by FRCA.

Compare this behavior to the HTTP Server (powered by Apache) local cache directive LiveLocalCache. LiveLocalCache checks to see if the file is updated in the IFS each time it is requested. If it is not updated, the file is served from the cache. If it is updated, then the entry for this file in the local cache is marked invalid and the file is served from the IFS for all subsequent requests. You have to restart your server to cause it to be loaded back in the local cache. See 10.3.1, “HTTP Server (powered by Apache) local cache” on page 236, for more information.

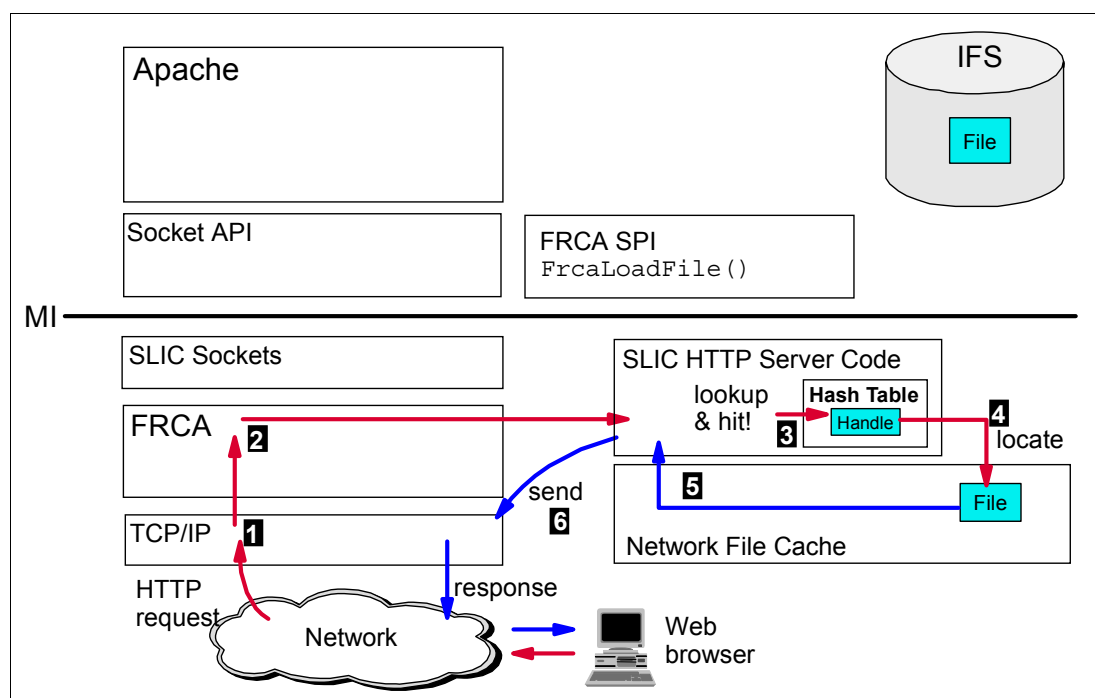


Figure 10-36 FRCA: Local cache hit scenario

10.6.3 How FRCA reverse proxy cache works

This section offers a series of diagrams that show the dramatic shift in processing from above the MI to below as we follow each client get request.

FRCA: Reverse proxy miss scenario

Figure 10-37 shows a FRCA reverse proxy miss scenario. That is, when FRCA recognizes that content for an incoming URI should be cached but is not.

The steps from request through response are:

1. An HTTP request is received by TCP and passed to FRCA.
2. FRCA uses the URI as part of the lookup key to see if this dynamic content is cached in the FRCA network proxy cache. It has not (miss!).
3. As part of the configuration of the FRCA reverse proxy, a new HTTP request is sent to the configured URL (for this URI). This dynamic content server (called an *origin server*) is contacted via TCP/IP. This origin server can be located on the same iSeries server or anywhere connected via TCP/IP.
4. The origin server returns the content.
5. FRCA caches the content and updates the hash table (for the next time).
6. The content is sent back to the Web browser.

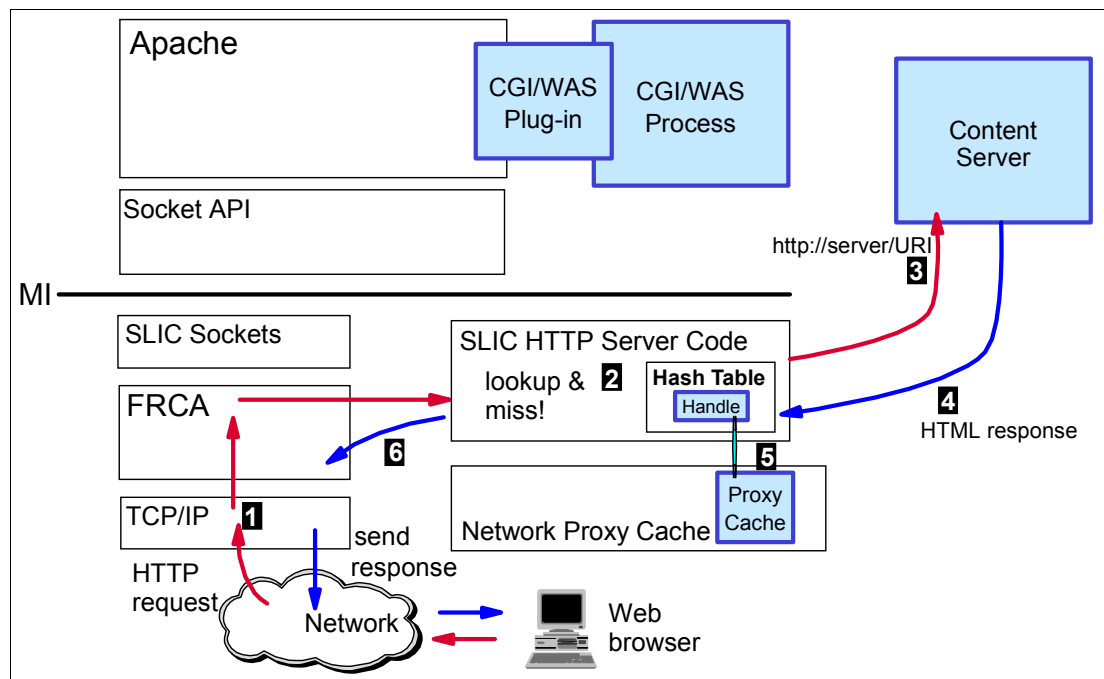


Figure 10-37 FRCA: Reverse proxy miss scenario

FRCA: Reverse proxy hit scenario

Figure 10-38 shows an FRCA reverse proxy hit scenario. The steps from request through response are:

1. An HTTP request is received by TCP and passed to FRCA.
2. FRCA uses the URI as part of the lookup key to see if this dynamic content is cached in the FRCA network proxy cache. It is (hit!).
3. FRCA reads the content in the network proxy cache.
4. FRCA sends the content back to the Web browser.

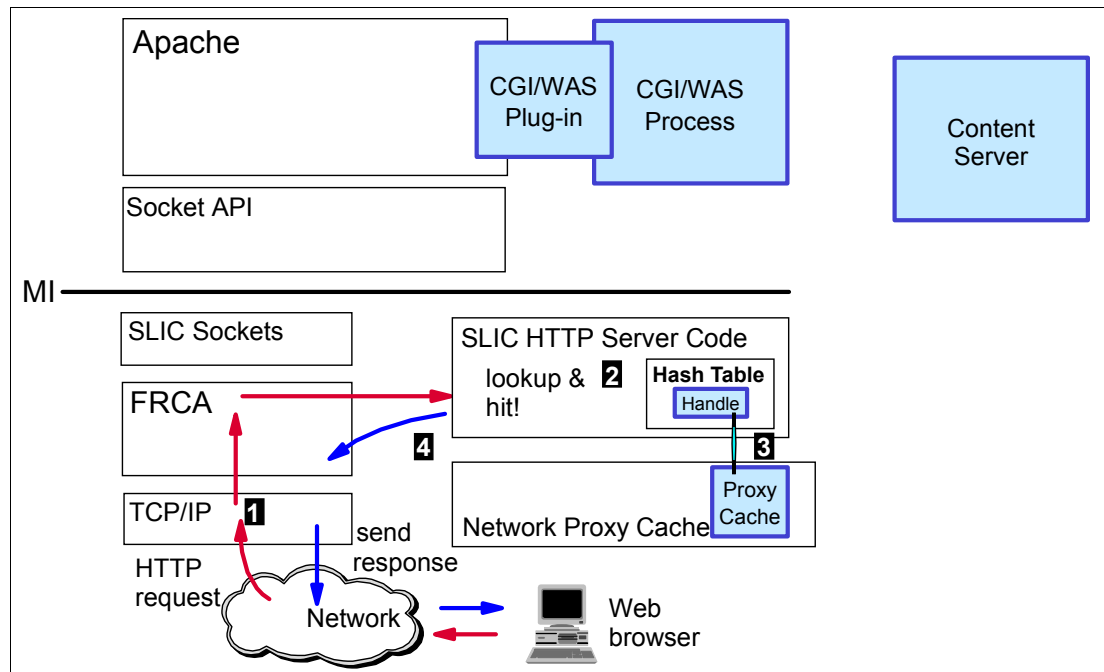


Figure 10-38 FRCA: Reverse proxy hit scenario

10.6.4 FRCA limitations

All the limitations mentioned in this section are the result of one reason. That is that FRCA is a SLIC task running below the MI. Therefore, it cannot take advantage of some of the server API as provided in OS/400.

- FRCA does not support SSL or TLS. Therefore you cannot enable FRCA cache for the sessions or ports with SSL/TLS. The reason is that applications (above the MI) write to the sockets API, which is currently unavailable to FRCA. FRCA can be configured for a non-secured port, such as port 80, for example, even in the same HTTP server that is also listening on a SSL encrypted port of 443.
- After the file loaded into the NFC, it can be accessed by any users accessing files in the same server instance. For this reason, you should enable the FRCA cache only for the contents that can be public. If some HTML files in IFS can be accessed only by authenticated users and one of the authenticated users has accessed such a file, the file is loaded in to the NFC by FRCA. Now when an unauthenticated user requests the same file, FRCA serves this file without user authentication.
- Similarly, since the code conversion is also performed above MI, code conversion is not supported. IFS files are read in binary and loaded into the cache as is. Generally, we don't use code conversion for IFS files so this limitation should have little impact.

Due to these limitations, the basic rule for FRCA is to use it to help serve the public portions of your Web application. Consider your public home page and all GIFs and JPEGs as an example. If and when the customer enters, a portion of your Web application that is secured with Basic Authentication or SSL/TLS turns off FRCA.

The presence of any of the following headers in an HTTP request forces FRCA to pass the request directly to the local HTTP Server (powered by Apache) or remote content server (via reverse proxy) without checking the cache:

- ▶ authorization
- ▶ allow
- ▶ cache-control
- ▶ content-base
- ▶ content-encoding
- ▶ content-language
- ▶ content-location
- ▶ content-md5
- ▶ content-range
- ▶ date
- ▶ etag
- ▶ expires
- ▶ if-match
- ▶ if-none-match
- ▶ if-range
- ▶ last-modified
- ▶ max-forwards
- ▶ proxy-authorization
- ▶ public
- ▶ protocol-request
- ▶ range
- ▶ retryafter
- ▶ transfer-encoding
- ▶ upgrade
- ▶ vary
- ▶ www-authenticate
- ▶ warning

10.6.5 FRCA configuration examples

Refer to the following examples of using FRCA:

- ▶ “Configuring FRCA for local cache” on page 288
- ▶ “Configuring FRCA for reverse proxy cache” on page 292
- ▶ “A more complete FRCA configuration example” on page 295

Before getting started with FRCA, configure the Network File Cache.

Network File Cache

FRCA local cache (only) uses the NFC component of OS/400. The NFC is a SLIC component that is basically a file system that allows other SLIC tasks to open, read, write, and close stream files. All this happens below the MI as shown in Figure 10-3 on page 227.

Starting from V5R2, by default, the NFC is enabled and has a size of 10 MB allocated out of the base user pool. Due to OS/400’s single-level store, even if you do not use the NFC, it simply pages out to direct access storage device (DASD) and it does not disturb the running of your other applications.

The size you define for the NFC is the maximum amount of storage available to all the individual instances of the HTTP Server (powered by Apache) that are configured to use FRCA combined. For example, you may have three instances of the HTTP Server (powered by Apache) that are using FRCA, each with the `FRCACacheLocalSizeLimit` set to 1 MB. In this case, configure the NFC to have a size of 3 MB.

You can change the parameters of the NFC using the Change TCP/IP Attributes (CHGTCPA) command. `CHGTCPA NFC(*YES 300 10)` changes the settings for the NFC to the defaults.

Tip: FRCA reverse proxy cache simply allocates main store memory out of the base user pool to hold the HTML results pages from the remote content servers. As shown in Figure 10-37 on page 285 and Figure 10-38 on page 286, this cache is named *network proxy cache*, although it really does not have a proper name. Generally, these HTML results pages are small. It was thought to be more efficient to maintain an in-memory cache, rather than to add the overhead of the NFC.

Configuring FRCA for local cache

This scenario uses FRCA local cache to cache all the files in a SiteMap/* subdirectory. Table 10-4 shows all the important characteristics of this Web application. This Web application ran in a shared environment in the International Technical Support Organization (ITSO), which explains the ports names such as 8000 (equivalent to port 80) and SSL-enabled port 44300 (equivalent to port 443).

Table 10-4 FRCA: HTTP Server (powered by Apache) used for local cache

Parameter	Value
Server name	PBABASIC00
Server root	/http53d00/basicConfig
Document root	/http53d00/basicConfig/ITSOco
Directory to be cached by FRCA local cache	/http53d00/basicConfig/ITSOco/SiteMap/*
IP address	All
Ports	8000 (FRCA enabled) 44300 (for SSL, not FRCA enabled)

We recommend that before you start to configure FRCA, test your Web application to verify it is working as expected. In our case, everyone has access to the files in the SiteMap/* subdirectory.

To configure FRCA local cache, follow these steps. In this first part, we enable FRCA for port 8000 only.

1. From the Server list, select you server.
2. From the Server area list, select **Global configuration**. This is where all FRCA configuration should take place.
3. In the left pane, under Server Properties, select **FRCA**.
4. Select the **General Settings** tab.

5. On the General Settings page (Figure 10-39), complete these steps:
 - a. Select the radio button for the port for which you want to enable FRCA. In this case, we are using port 8000 for the public traffic and port 44300 for the SSL encrypted traffic.
 - b. Under the FRCA column, select **Enabled**. This configuration feature allows you to enable, by port, FRCA caching.
 - c. Click **Continue** to keep the changed configuration and to stay on this form.

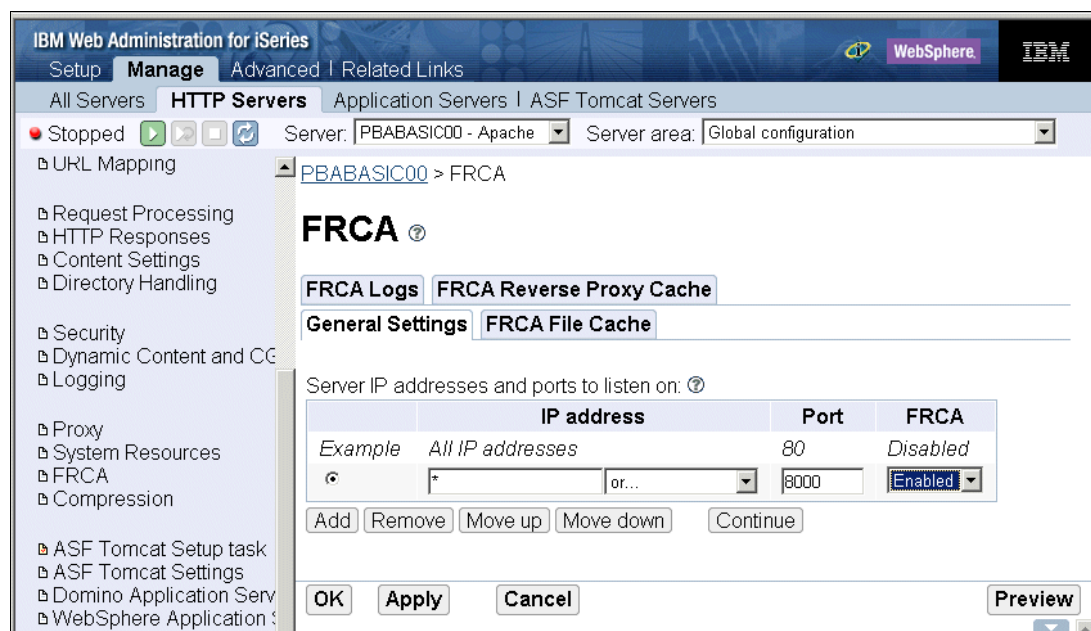


Figure 10-39 FRCA: Enabled for a specific port

6. Configure the FRCA local cache to cache all the contents of a Directory context. Select the **FRCA File Cache** tab.
7. On the FRCA File Cache page (Figure 10-40), complete these tasks:
 - a. For FRCA file cache capabilities, select **Enabled**. This option allows you to turn FRCA local cache on and off with ease, without having to resort to commenting out or deleting all the FRCA local cache configuration directives.
 - b. Optional: Restrict the maximum size of the FRCA local cache and define the maximum file size to be cached with these parameters. To keep our example simple, we keep them as the defaults.
 - c. Under Files to cache during server startup, click **Add**.
 - d. A new row is added to the table in which you can enter the file path and names that you want FRCA to cache. For our example, we typed SiteMap/*. Here are some comments about FRCA directives:
 - Since SiteMap/* does not have a leading slash (/), we assume it to be relative to Document root.
 - FRCA configuration directives are case sensitive, unlike Apache configuration directives.
 - FRCA configuration directives are not recursive, unlike Apache configuration directives. That is, in this situation you are telling FRCA to cache files in the SiteMap/* directory only. If you need to cache multiple subdirectories or multiple file types, you have one FRCA directive for each.

Tip: You can choose to have FRCA local cache files during server runtime or at startup. The choice is a subtle one, however. Both choices behave in a similar way in that the first request for a file in the SiteMap/* subdirectory is served by the HTTP Server (powered by Apache) and the second and all subsequent requests are served by FRCA.

Caching at server startup requires some additional CPU at server startup time to ready FRCA for caching the file at the first request. When the first request is made for the file and served from the HTTP Server (powered by Apache), the CPU time it takes to load the NFC with the file is reduced.

Caching at server runtime requires less CPU at server startup time. When the first request is made for the file and served from the HTTP Server (powered by Apache), the CPU time it takes to load the NFC with the file is greater than if you selected to cache at server startup.

- e. Click **Continue**. If you have more files or file types that you want to cache, repeat step c.
- f. Click **OK** to save all your configuration changes.

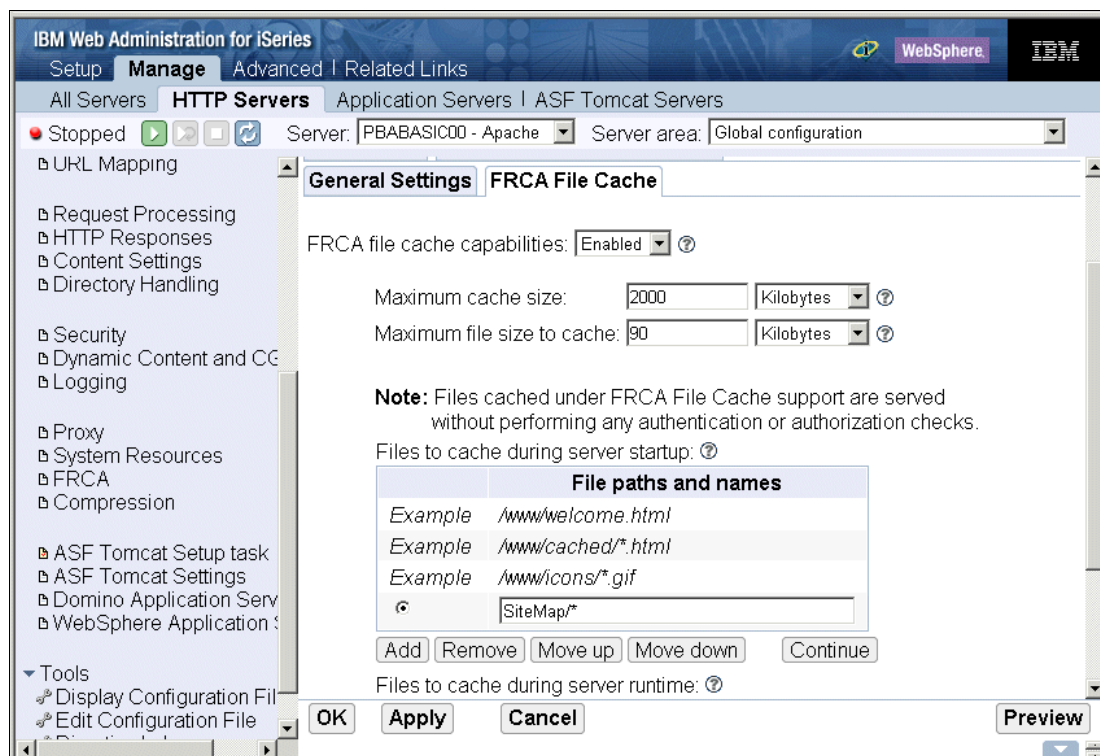


Figure 10-40 FRCA: Configuring for caching the contents of a Directory context

If you follow these steps, you have these three directives in your HTTP Server (powered by Apache) configuration file:

```
Listen *:8000 FRCA
FRCAEnableFileCache On
FRCACacheLocalFileStartUp SiteMap/*
```

And, finally test your server. It is a bit difficult to tell whether your HTTP Server (powered by Apache) served a file or if it was done by FRCA. The end result to the Web client is the same, of course. One sure way is to use Start Communications Trace (STRCMNTRC) command.

The HTTP Server (powered by Apache) may return an HTTP response that looks something like the following example (the key text is highlighted in bold):

```
*.....E...*G...@.*K.*.*.**P"*.**.*
*.*.*P. .DN..HTTP/1.1 200 OK..DA*
*TE: SAT, 03 AUG 2002 13:15:47 GM*
*T..SERVER: APACHE..LAST-MODIFIED*
*: SAT, 03 AUG 2002 01:37:39 GMT.*
*.ETAG: "4B33-3CB-C07A7EC0"..ACCE*
*PT-RANGES: BYTES..CONTENT-LENGTH*
*: 971..KEEP-ALIVE: TIMEOUT=15, M*
*AX=100..CONNECTION: KEEP-ALIVE..*
*CONTENT-TYPE: TEXT/HTML; CHARSET*
*=ISO-8859-1....<HTML>.FRI AUG 02*
```

FRCA local cache returns this (the key text is highlighted in bold):

```
*.....E...*J...@.*.*.*.*.**P"*.**.*
*@***P..-***..HTTP/1.1 200 OK..DA*
*TE: SAT, 03 AUG 2002 13:15:54 GM*
*T..SERVER: APACHE/2.0.43(FRCA)..*
*ACCEPT-RANGES: BYTES..CONNECTION*
*: KEEP-ALIVE..LAST-MODIFIED: SAT*
*, 03 AUG 2002 13:15:48 GMT..CONT*
*ENT-TYPE: TEXT/HTML..CONTENT-LEN*
*GTH: 971..X-CACHE: HIT FROM APAC*
*HE/2.0.43(FRCA)....<HTML>.FRI AU*
```

Tip: It may look like the FRCA server is based on or uses code borrowed from ASF to provide this HTTP server function below the MI. This is not the case, however. FRCA is a mini-HTTP server that has been written by IBM specifically for this purpose.

You can find more details about communications trace in 13.2.9, “Communications trace” on page 353. To test FRCA on your iSeries using communications trace, follow these steps:

1. Start your HTTP Server (powered by Apache) instance.
2. Enter the Start Communications Trace (STRCMNTRC) command.
3. Request a file from the SiteMap/* subdirectory from a Web client. This first request is served from the HTTP Server (powered by Apache) since FRCA stores this file in the NFC.
4. Clear the local cache from your Web client. Microsoft Internet Explorer has a habit of caching files when you least expect it. For Internet Explorer, select **Tools → Internet Options → Delete Files**.
5. Request the same file again from SiteMap/* should cause FRCA to serve that file directly from the NFC.
6. Enter the End Communications Trace (ENDCMNTRC) command.
7. Enter the Print Communications Trace (PRTCMNTRC) command. Then open the spooled trace file. Searching for “(FRCA)” usually finds the place where FRCA served the file.
8. Make sure to enter the Delete Communications Trace (DLTCMNTRC) command for the next time you want to start a communications trace.

Configuring FRCA for reverse proxy cache

Dynamic content, such as result of an HTTP request to a content server, can be cached by specifying a URI to identify the request that is then mapped to a fully qualified URL. This is reverse proxy cache support that allows you to access an HTTP server either on this same iSeries or anywhere on your intranet (or Internet) to provide dynamic content that is automatically cached in FRCA. A timer is used to determine when cached items are stale. An example is:

```
FRCAProxyPass /cgi-bin/ http://as20.domain.com:9999/cgi-bin/
```

For this scenario, we want to cache a document that is generated dynamically by a Net.Data macro (this is our content server) each time a Web client requests that page. Net.Data is a good scripting language from IBM that is similar to PHP in the Apache world and JavaServer Pages (JSPs) and servlets in the Java 2 Platform, Enterprise Edition (J2EE) world. But, if your iSeries has to serve hundreds, if not thousands, of these Net.Data generated pages, you may want to find a way to cache the results.

We assume that the Web server PBABASIC00 defined in Table 10-4 on page 288 will perform the function of our content server. We create a new HTTP Server (powered by Apache) based on the values in Table 10-5.

Table 10-5 FRCA: HTTP Server (powered by Apache) used for reverse proxy cache

Parameter	Value
Server name	PBAFRCA00
Server root	/http53d00/FRCA
Document root	/http53d00/FRCA/ITSOco
IP address	All
Port	8600
If incoming URI is...	/cgi-bin/MACRO1.MBR/
Send request to content server URL	http://as20:8000/cgi-bin/MACRO1.MBR/
Content Server	Server PBABASIC00 as defined by Table 10-4 on page 288

Nothing can stop you from doing all this work in the same Apache instance on your iSeries server. It is easier to conceptualize as two different servers. PBABASIC00 is the remote content server, and PBAFRCA00 is used as a front end to the remote content server.

To configure FRCA reverse proxy cache, follow these steps. The first step is to enable FRCA for port 8600. The configuration steps for this new server PBAFRCA00 are similar to those shown in Figure 10-39 on page 289.

1. From the Server list, select your server. For Server area, select **Global configuration**. This is where all FRCA configuration should take place.
2. In the left pane, under Server Properties, select **FRCA**.
3. Select the **General Settings** tab.
4. Select the radio button for the port for which you want to enable FRCA. In this case, we are using port 8600. Under the FRCA column, select **Enabled**.
5. Click **Continue** to keep the changed configuration and to stay on this form.

6. Enable FRCA for reverse proxy cache as shown in Figure 10-41.
 - a. Select the **FRCA Reverse Proxy Cache** tab.
 - b. For the FRCA reverse proxy cache capabilities, select **Enabled**. This option allows you to turn FRCA reverse proxy cache on and off with ease, without resorting to commenting out or deleting all the FRCA reverse proxy cache configuration directives.
 - c. Optional: Restrict the maximum size of the FRCA reverse proxy cache and define the maximum file size to be cached with these parameters. To keep our example simple, we keep them as the defaults.
 - d. Under Proxy requests to remote servers, click **Add**. A new row is added to the table in which you can enter the local virtual path (URI) and the remote server URL. For our example, we entered `/cgi-bin/MACRO1.MBR/` and `http://as20:8000/cgi-bin/MACRO1.MBR/` respectively. The same comments about FRCA directives apply here as they did for FRCA local cache.

In addition, FRCA reverse proxy cache handles the incoming URI in the same way you expect any reverse proxy cache. The matching URI text is stripped off. Additional text to the right is saved and appended to the end of the fully qualified remote server URL. As you can see in our example, we repeat the string `/cgi-bin/MACRO1.MBR/` as part of the remote server URL so we do not lose this information.

 - e. Click **Continue**.

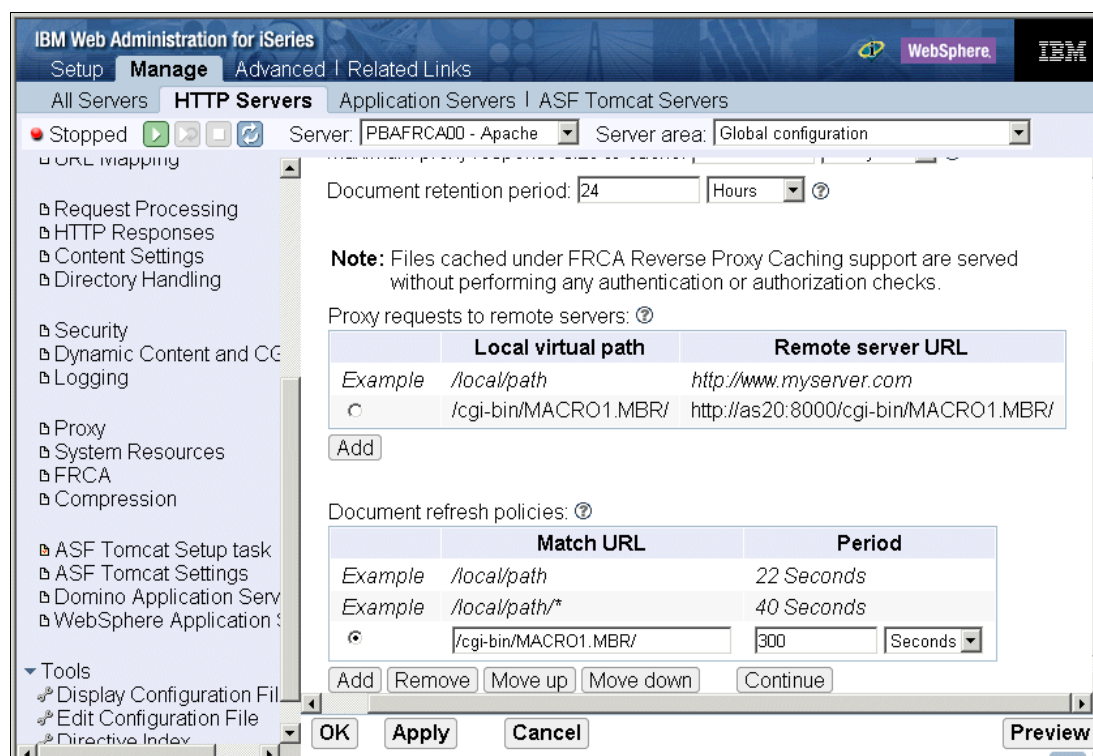


Figure 10-41 FRCA: Configuring for reverse proxy caching

- f. Under Document refresh policies, click **Add**. Type `/cgi-bin/MACRO1.MBR/` for Match URL and 300 seconds for Period. In this example, we tell FRCA that the proxy cache item will expire when the minimum of the `FrcaProxyCacheExpiryLimit` and the HTTP response expiration time is reached. FRCA reverse proxy only caches responses that do not contain headers that prohibit caching (that is `MaxAge=0`).

The response is cached the first time it is received from the content server and then is continually refreshed at the specified interval. If multiple responses are cached by the same FrcaProxyCacheRefreshInterval, the refresh is distributed evenly across the specified refresh interval, to prevent all of the responses from being refreshed at the same time. The main advantage of the FrcaProxyCacheRefreshInterval is that once a response is cached, a valid copy always exists in the proxy cache. There is never a need to wait for the response to be retrieved from the content server.

- g. Click **Continue**.
- h. Click **OK** to save all your configuration changes.

If you follow these steps, you change and add four directives in your HTTP Server (powered by Apache) configuration file:

```
Listen *:8600 FRCA
FRCAEnableProxy On
FRCAProxyPass /cgi-bin/MACR01.MBR/ http://as20:8000/cgi-bin/MACR01.MBR/
FRCAProxyCacheRefreshInterval /cgi-bin/MACR01.MBR/ 300
```

Finally, test your server. In this case, two instances of the HTTP Server (powered by Apache) are running. To determine if FRCA is caching the HTML result pages from our content server, simply monitor the iSeries thread that is handling the Net.Data macro invocation in the PBABASIC00 server. That is, regardless of the number of times FRCA reverse proxy serves the HTML results page, the CPU seconds for the content server should not increase.

On your iSeries server, follow these steps:

1. Start both of the HTTP Server (powered by Apache) instances PBABASIC00 and PBAFRCA00.
2. From a 5250 session, enter the Work with Active Jobs (WRKACTJOB) command to find the tasks associated with the remote content server PBABASIC00. It should look similar to the example in Figure 10-42.

Opt	Subsystem/Job	User	Type	CPU %	Function	Status
	PBABASIC00	QTMHHTTP	BCH	.0	PGM-QZHBHTTP	SIGW
	PBABASIC00	QTMHHTTP	BCI	.0	PGM-QZSRLOG	SIGW
	PBABASIC00	QTMHHTTP	BCI	.0	PGM-QZSRLOG	SIGW
	PBABASIC00	QTMHHTTP	BCI	.0	PGM-QZSRHTTP	SIGW
	PBABASIC00	QTMHHTTP	BCI	.0	PGM-QZSRHTTP	SIGW
12	PBABASIC00	QTMHHTTP	BCI	.0	PGM-QZSRCGI	TIMW

Figure 10-42 WRKACTJOB: Displaying the list of threads; PGM-QZSRCGI is the CGI thread

3. Enter option 12 (Work with threads) next to the BCI job running as a CGI as indicated by the function PGM-QZSRCGI. As shown in Figure 10-43, you should see a single line that indicates how many CPU seconds this thread has used.

Opt	Thread	Status	Total CPU	Aux I/O	Run Priority
	0000002E	TIMW	.521	484	25

Figure 10-43 WRKACTJOB: Option 12 (Work with threads)

4. From your Web client, enter the URL:
<http://as20:8600/cgi-bin/MACR01.MBR/run>

Port 8600 is the HTTP Server (powered by Apache) instance PBAFRCA00. Your FRCA reverse proxy configuration redirects this request to the PBABASIC00 instance at port 8000 on the same iSeries server.

5. Back on your 5250 session, press F5 (Refresh). You should see the Total CPU seconds for this thread handling the Net.Data CGI invocation.
6. Back on your Web client, click **Refresh** (which is also F5). In fact, click Refresh as many times as you want, clear the client cache, or do what every you want. You do not see the Net.Data task on the iSeries spend any CPU because FRCA is serving the resulting HTML from its cache.

A more complete FRCA configuration example

As shown in Figure 10-44 on page 296, this example is more complex and complete of FRCA configuration for a Web application. This example includes:

- ▶ Static HTML and GIF content as served from the subdirectory contexts of /Downloads and /People.
- ▶ Dynamic content from:
 - A WebSphere Application Server content server running on the local iSeries server defined as as20.itsoroch.ibm.com
 - A CGI content server running on a remote iSeries server defined as as21.itsoroch.ibm.com

We do not show you all the steps to configure FRCA via the administration GUI, but instead offer a description of the FRCA directive and its effect on your Web application. The following numbers correspond to the line number of the directives found in the httpd.conf configuration file in Figure 10-44. Figure 10-44 also shows pairs of numbers. The configuration directives in the lower part of this figure match the functional diagram in the upper part of the figure.

2 Listen 10.5.92.14:8080 FRCA

Specifying the Listen directive with the parameter FRCA enables FRCA cache for this port.

5 FRCAEnableFileCache On

This directive enables FRCA cache for this server instance ITS099. The other directives for specific settings of FRCA all depends on this directive is on or off.

11 FRCACacheLocalFileStartUp /ITS0/itso99/ITS0co/Downloads/*.html

By specifying this directive, the files that have an .html extension in the directory /ITS0/itso99/ITS0co/Downloads are all cached when you start the server ITS099.

12 FRCACacheLocalFileRunTime /ITS0/itso99/ITS0co/People/*

This directive makes all files in the directory /ITS0/itso99/ITS0co/People available to be cached when they are accessed. In this example, the files in the subdirectory /Employees are not cached because file name matching is not recursive.

15 FRCAEnableProxy On

This directive enables FRCA proxy.

16 FRCAProxyPass /servlet/ http://10.5.92.14:8080/servlet/

In this example, specifying /servlet in URI causes it to run a servlet on the application server. By specifying the directive FRCAProxyPass as in this example, the result of the servlet can be cached in the NFC for a certain period, that is specified by the directive FRCAProxyCacheRefreshInterval.

In this example, the target URL has the same IP address and port as the ones on which this server listens. In this case, FRCA recognizes that this URL is for the same server and passes the request to the HTTP Server (powered by Apache) without any looping problem.

17 `FRCAProxyCacheRefreshInterval /servlet/ 300`

As described above, this directive specifies the interval of refreshing cached data of FRCA proxy.

20 `FRCAProxyPass /cgi-bin/ http://as21.itsoroch.ibm.com:9999 /cgi-bin/`

By specifying this directive, the request for CGI program is rerouted to the remote iSeries as21.itsoroch.ibm.com:9999. The result is cached in the NFC on as20.itsoroch.ibm.com.

21 `FRCAProxyCacheRefreshInterval /cgi-bin/ 180`

As described earlier, this directive specifies the interval of refreshing cached data of FRCA proxy.

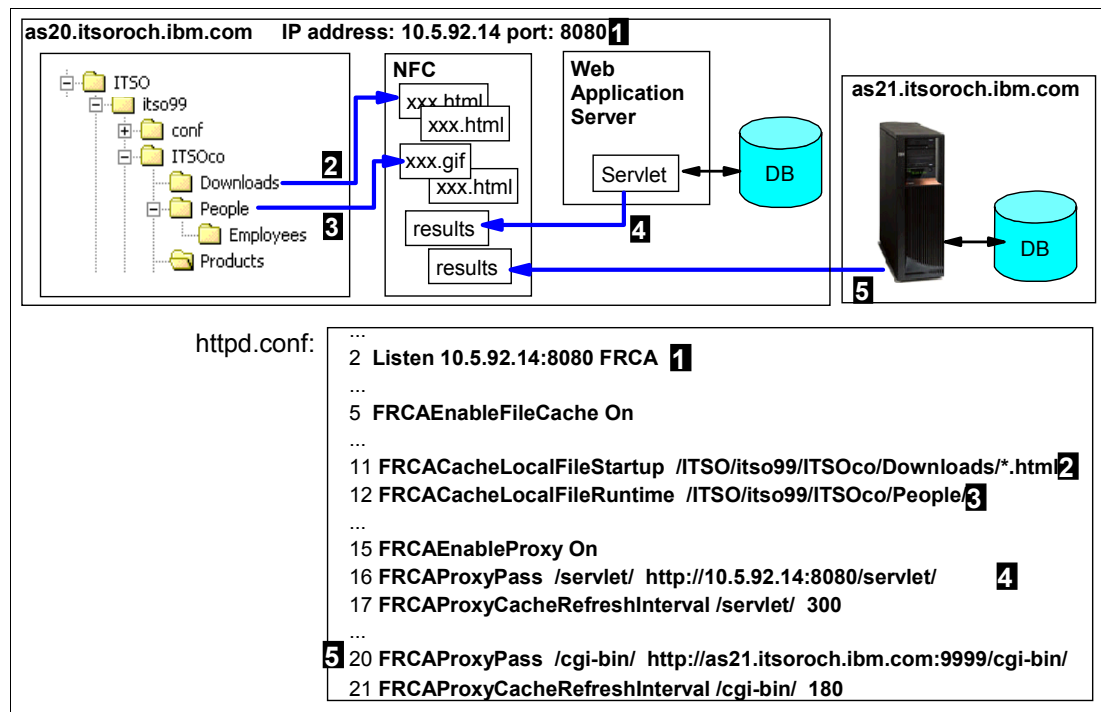


Figure 10-44 FRCA: Complete configuration example

10.6.6 Miscellaneous FRCA directives beyond the online help

Most FRCA directives are defined in the online help associated with your HTTP Server (powered by Apache). While you are managing the details of any Apache server, you can click Directive Index under Tools. All the FRCA directives, except Listen, start with "FRCA" so you can find them easily. Here is a list to get you started:

```

FRCACacheLocalFileRunTime
FRCACacheLocalFileSizeLimit
FRCACacheLocalFileStartUp
FRCACacheLocalSizeLimit
FRCACustomLog
FRCAEnableFileCache
FRCAEnableProxy
FRCAProxyCacheEntitySizeLimit

```



```
FRCAProxyCacheExpiryLimit
FRCAProxyCacheRefreshInterval
FRCAProxyCacheSizeLimit
FRCAProxyPass
Listen (with the FRCA parameter)
```

Some, however, are not defined in the GUI online help text. These are:

```
FRCACookieAware
FRCAEndofURLMarker
FRCAMaxCommBufferSize
FRCAMaxCommTime
FRCARandomizeResponse
```

As far as we can tell, these directives are not formally documented anywhere. They are used to perform specific services to dramatically improve the performance of FRCA. These FRCA directives are documented in the following sections.

FRCACookieAware

Syntax:

```
FRCACookieAware <path>
```

Example:

```
FRCACookieAware /some_path_segment
```

This FRCA directive indicates a URL prefix for which the cookie should be included in cache lookup. This directive makes it possible to serve a cached entity only for the requests with the same cookie. This allows content that is intended for specific individuals to be cached separately.

FRCAEndofURLMarker

Syntax:

```
FRCAEndofURLMarker <marker>
```

Example:

```
FRCAEndofURLMarker ###
```

FRCA support can identify the end of the original URL (link) before it is modified or padded by the client. Specify the unique string that identifies the end of URLs. Suppose a link in an HTML page is:

```
http://some.org/some_path/some_parms###
```

Before a client sends this request to the server, it may pad the URL with some data such as `client_padded_data`. In this case, your HTTP Server (powered by Apache) receives the path:

```
/some_path/some_parms###client_padded_data
```

Specify the following directive:

```
FRCAEndofURLMarker ###
```

FRCA support can identify the end of the original URL (link) before it is modified (padded) by the client.

FRCAMaxCommBufferSize

Syntax:

```
FRCAMaxCommBufferSize <bytes>
```

Example:

```
FRCAMaxCommBufferSize 4000000
```

This directive sets the communication buffer size (in bytes) in FRCA for performance. The data being sent to the HTTP Server (powered by Apache) consists of log data, message data, and collection services data. FRCA buffers the size of data specified until the buffer is full. After the buffer is full, the data is transmitted to the HTTP Server (powered by Apache) for processing.

FRCAMaxCommTime

Syntax:

```
FRCAMaxCommTime <seconds>
```

Example:

```
FRCAMaxCommTime 240
```

This directive sets the maximum number of seconds to wait before the data buffer is sent from FRCA to the HTTP Server (powered by Apache). The data being sent to the HTTP Server (powered by Apache) consists of log data, message data, and collection services data. After the time limit is reached, the data is transmitted to the HTTP Server (powered by Apache) for processing.

FRCARandomizeResponse

Syntax:

```
FRCARandomizeResponse <path> <string> <nnn> <mmm>
```

Note the following explanation:

- ▶ **<path>**: A valid path in the form:

```
/some_path_segment/some_partial_file_nameNNN.ext
```

The NNN marker is replaced with a randomly generated whole number by FRCA before serving the response.

- ▶ **<string>**: The replacement string marker (NNN) in the path.
- ▶ **<nnn>**: Lower bound of the random numbers (whole number integers) that FRCA generates.
- ▶ **<mmm>**: Upper bound of the random numbers (whole number integers) that FRCA generates.

Examples:

```
FRCARandomizeResponse /some_path/fileNNN.html NNN 1 999
```

```
FRCARandomizeResponse /some_path/fileXXX.html XXX 200 300
```

Specify the path template, the replacement string marker, and the random number range that you want FRCA to use to randomly select and serve files of that template.

The most likely place this is used is on your home page. Normally, FRCA caches the content of your home page for a configurable period of time. During that period of time, your advertising banner is the same for everybody. It does not change. This configuration directive allows you to create NNN home pages (index.html) and have FRCA randomly select from its cache which one to send as an individual response to each and every request. Even the same client using the refresh key randomly receives a different advertising banner.

For example, if you have 999 “advertising” files with the names file1.html through file999.html in your server document root, then configure:

```
FRCARandomizeResponse /document_root_alias_path/fileNNN.html NNN 1 999
```

Then request the URL:

```
http://some_host:port/dirpath/fileNNN.html
```

FRCA randomly selects and serves one of the 999 files.

10.6.7 The FRCA challenge

You now can see how easy it is to take advantage of FRCA’s local and reverse proxy caching. It is simply a matter of configuration. FRCA really can be thought of as providing both a local cache for public documents that tend to be static and reverse proxy for documents that tend to be dynamic. For the FRCA challenge, use:

- ▶ FRCA local cache to cache graphic files (GIFs, JPEGs, and so on). These graphic files tend to be the caching world’s equivalent of “low-hanging fruit” for a Web application.
- ▶ FRCA reverse proxy cache to cache just a single dynamic page: your index.html, or home page. This single HTML page is usually the most popular page and most likely is thick with dynamic content.

Over time, you will become more comfortable with FRCA on your iSeries HTTP Server (powered by Apache). Then you can expand the range of items cached with a bit of simple configuration and testing.

10.6.8 For more information

Here are some pointers to more information about FRCA:

- ▶ *iSeries Performance Capabilities Reference Version 5, Release 3*, SC41-0607
<http://www-1.ibm.com/servers/eserver/iseries/perfmgmt/resource.htm>
- ▶ Detailed reference documentation on FRCA and its configuration directives, which is located in the iSeries Information Center at:
<http://publib.boulder.ibm.com/html/as400/infocenter.html>
- ▶ The online help associated with your HTTP Server (powered by Apache), which defines most of the FRCA directives. While you are managing the details of any Apache server, click **Tools** → **Directive Index**. All the FRCA directives start with “FRCA” so you can find them easily.

10.7 Cryptographic coprocessors

IBM offers two cryptographic hardware solutions for customers that require a high level of security for data stored on their iSeries server and for SSL/TLS transactions. These options have a lot to offer iSeries server customers, including enhanced SSL/TLS performance. IBM offers the following cryptographic hardware options:

- ▶ IBM 2058 e-business Cryptographic Accelerator (hardware feature code: 4805)
- ▶ IBM 4758 Cryptographic Coprocessor (hardware feature codes: 4801 or 4802)

The benefits of each cryptographic hardware option include:

- ▶ IBM 2058 e-business Cryptographic Accelerator:
 - Offers a large capacity for accelerating SSL transactions
 - Minimal installation and configuration effort
 - Minimal management requirements
- ▶ IBM 4758 e-business Cryptographic Coprocessor:
 - Tamper-resistant hardware features
 - Numerous configuration options, enabling you to customize functions to fit your needs
 - Provides secure key storage for applications and SSL transactions
 - Offers a rich set of cryptographic functions for applications, including Triple-DES, RSA digital signature support, financial PIN processing, and robust key management services

These cryptographic accelerator coprocessors are used by the system on an SSL/TLS connection handshake process (negotiates the level of SSL support and key exchange information to be used by an SSL session). Your configuration and use of the DCM and client or server digital certificates does not change.

iSeries Performance Capabilities Reference Version 5, Release 3, SC41-0607, has test results that show the handshaking performance improvements for both the older 4758 technology and the new technology 2058. The 2058 supports the same algorithms as the 4758. The 2058 does not support any secure key usage or key management. See the following Web site for more information:

<http://www-1.ibm.com/servers/eserver/iseries/perfmgmt/resource.htm>

Tip: The iSeries already held the number three spot on the SPECweb99 and the number two spot on the SPECweb99 SSL benchmarks. This is a significant validation of the overall systems performance of the entire iSeries server. It is the iSeries' balanced ability to scale and run enormous On Demand Business workloads that is the basis for these (and other) benchmark successes. Note that FRCA and the cryptographic accelerator coprocessors were not used for the iSeries SPECweb99 SSL benchmark. FRCA could not due to an architectural limitation. The 4758 and 2058 cryptographic accelerator coprocessors could not since the SPECweb99 SSL benchmark does not allow hardware assist.

And, our ability to run enormous On Demand Business workloads is due to the integration of the SSL and TLS component 5722-AC3, the HTTP Server (powered by Apache) integration with OS/400. It is also due to the pure power of the iSeries' 64-bit RISC POWER processors, which allow the iSeries to climb to near the top of these benchmarks, even without using the cryptographic accelerator coprocessors.

You can find detailed information about these features in the V5R3 Information Center, under **Security**. You should also refer to the redbook *IBM @server iSeries Wired Network Security: OS/400 V5R1 DCM and Cryptographic Enhancements*, SG24-6168.

10.8 Real Time Server Statistics

This function was introduced as PTF for OS/400 V5R1 and V5R2 and was implemented with i5/OS V5R3 in the base version. Besides other enhancements, forms on the HTTP administrative GUI are provided for displaying HTTP Server statistics that are recorded for collection services (similar to the Original server monitor form).

The Real Time Server Statistics form and tabs in the IBM Web Administration for iSeries GUI provide information about server performance. You can only view statistics for running servers. You may choose this form to automatically refresh every 10 or 30 seconds, or 1, 5 or 10 minutes, by changing the Refresh Interval in the upper half of the right panel. The default is to refresh the data manually using the Refresh button located at the bottom of the form.

As shown in Figure 10-45, the upper part always describes the server, when it started, the current date and time, and the period of time the data has been measured. The lower part of the Real Time Server Statistics form contains five tabs. Each contains a different set of statistical information.

The type of information displayed depends on the activity of the HTTP Server (powered by Apache) and the functions that are enabled. Only statistical information for enabled or active functions are displayed. Each column heading identifies what enabled function or associated server is being surveyed for statistical information.

Tip: The Real Time Server Statistics provide real-time information about the running server environment. The information can help you determine if your server is set up correctly or if it needs some performance tuning. Following are a few examples that show how the statistics can help you detect configuration or performance problems:

- ▶ One piece of information that can help in several ways is the number of error responses. In today's complex Web content structure with many interlinked pages, it is likely that links may be broken and a user may receive a 404 (file not found) error message. Whenever this or other errors occur, the error response counter is updated. For example, if you see over a period of time a huge number of error responses, you know that something strange happens on your server. You can then go to the error log to determine the problems.
- ▶ The data provided for non-cache and cache responses tell you at a glance whether your cache configuration is working efficiently. For example, you may have decided to cache all HTML and GIF files from a certain directory, because you expected a huge number of hits for these resources. However, it turns out that the cached responses are quite low and the number of non-cached responses very high. In that case, you know that you cached the wrong resources and that you should use a Web analyzer to evaluate your access logs to see what resources were requested most.

Statistical information is cumulative. If a value is greater than $2^{64}-1$ in any column, the value resets to 0. All values reset to 0 if the server is stopped and then started.

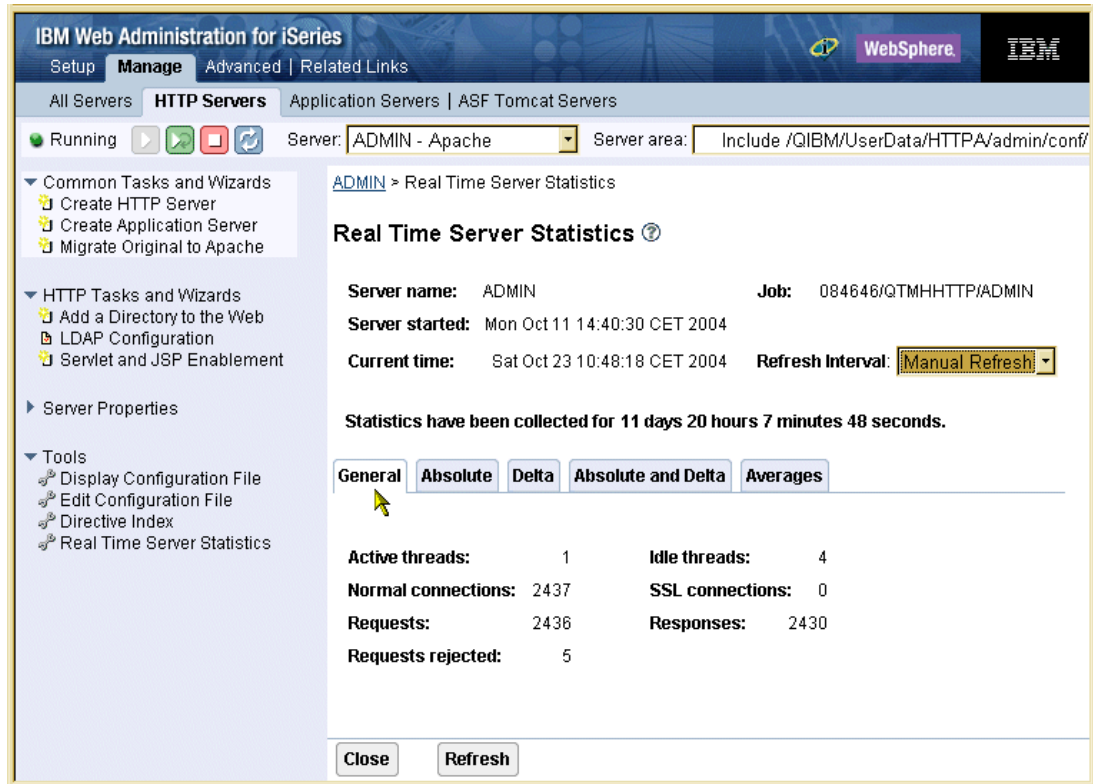


Figure 10-45 IBM Web Administration for iSeries: Real Time Server Statistics

Statistical information may be obtained for the following functions:

- ▶ **Server handled:** This column displays the number of completed server transactions by the server since the server was started. For example, it includes the completed transactions for static HTML pages, HTML pages containing SSIs, and images.
- ▶ **Proxy:** This column displays the number of completed server transactions that used proxy since the server was started. Proxy statistics are only available if proxy is enabled.
- ▶ **CGI:** This column displays the number of completed server transactions that were handled as CGI since the server was started. CGI statistics are available only if CGI is enabled.
- ▶ **Using SSL:** This column displays the number of completed server transactions that used SSL since the server was started. SSL statistics are available only if SSL is enabled.
- ▶ **WebSphere:** This column displays the number of completed server transactions that used an associated application server since HTTP Server (powered by Apache) was started. If the associated application server is not running, the information is still displayed but equals 0. WebSphere statistics are available only if a WebSphere Application Server is associated with an HTTP Server (powered by Apache).
- ▶ **Tomcat:** This column displays the number of completed server transactions that used an in-process or out-of-process ASF Tomcat server since HTTP Server (powered by Apache) was started.
- ▶ **Customer module:** This column displays the number of completed server transactions that used a customer or third-party module.
- ▶ **FRCA Stats:** This column displays the number of completed server transactions that used FRCA since the server was started. FRCA statistics are available only if FRCA is enabled.

- **FRCA Proxy:** This column displays the number of completed server transactions that used the FRCA proxy since the server was started. FRCA statistics are only available if FRCA is enabled.

The General tab as shown in Figure 10-45 displays basic information about the active server since the server was started. Statistical information displayed includes:

- **Active threads:** This field displays the number of currently active threads on the server. Active threads are typically processing a server request, such as serving a static page, calling a CGI script, or routing data to an application server.
- **Idle threads:** This field displays the number of currently idle threads active on the server. An idle thread is a portion of a program that is waiting for either a response or a request before it can continue. Idle threads are most often waiting for an HTTP request to process.
- **Normal connections:** This field displays the number of total normal (non-secure) connections to the server.
- **SSL connections:** This field displays the number of total SSL (secure) connections currently active.
- **Requests:** This field displays the number of total requests to the server since the server was started.
- **Responses:** This field displays the number of total responses from the server since the server was started.
- **Requests rejected:** This field displays the number of total rejected requests issued by the server since the server was started.

The Absolute and Delta tabs display statistical information about currently enabled functions or associated servers. The absolute value, as shown in our example in Figure 10-46, is a measurement of the total transactions since the server was started.

General

Absolute

Delta

Absolute and Delta

Averages

Detailed server statistics:

Type	Server handled	CGI	Tomcat	Customer module	Total
Requests	1463	59	966	0	2488
Responses	1463	59	966	0	2488
Error responses	0	0	0	0	0
Non-cache responses	1463	59	966	0	2488
Cache responses	0	0	0	0	0
Bytes received	1429097	88034	1057702	0	2574833
Bytes sent	626098	1372104	6708125	0	8706327
Non-cache Processing (seconds)	4.12	62.028	337.906	0.0	404.054
Cache Processing (seconds)	0.0	0	0	0	0.0

Figure 10-46 Real Time Server Statistics: Absolute tab

The delta value shown in Figure 10-47 is a measurement of the total transactions since the server statistics were refreshed.

General Absolute Delta Absolute and Delta Averages					
Detailed server statistics:					
Time since last refresh: 560 seconds.					
Type	Server handled	CGI	Tomcat	Customer module	Total
Requests	+0	+0	+1	+0	+1
Responses	+0	+0	+1	+0	+1
Error responses	+0	+0	+0	+0	+0
Non-cache responses	+0	+0	+1	+0	+1
Cache responses	+0	+0	+0	+0	+0
Bytes received	+0	+0	+1237	+0	+1237
Bytes sent	+0	+0	+12046	+0	+12046
Non-cache Processing (seconds)	+0.00	+0.000	+0.000	+0.0	+0.000
Cache Processing (seconds)	+0.0	+0	+0	+0	+0.0

Figure 10-47 Real Time Server Statistics: Delta tab

The absolute and delta statistical information may be displayed separately or side by side for comparison as in Figure 10-48.

Detailed server statistics:					
Time since last refresh: 275 seconds.					
Type	Server handled	CGI	Tomcat	Customer module	Total
Requests	1463 (+0)	59 (+0)	968 (+1)	0 (+0)	2490 (+1)
Responses	1463 (+0)	59 (+0)	968 (+1)	0 (+0)	2490 (+1)
Error responses	0 (+0)	0 (+0)	0 (+0)	0 (+0)	0 (+0)
Non-cache responses	1463 (+0)	59 (+0)	968 (+1)	0 (+0)	2490 (+1)
Cache responses	0 (+0)	0 (+0)	0 (+0)	0 (+0)	0 (+0)
Bytes received	1429097 (+0)	88034 (+0)	1060173 (+1234)	0 (+0)	2577304 (+1234)
Bytes sent	626098 (+0)	1372104 (+0)	6732308 (+12137)	0 (+0)	8730510 (+12137)
Non-cache Processing (seconds)	4.12 (+0.00)	62.028 (+0.000)	337.906 (+0.000)	0.0 (+0.0)	404.054 (+0.000)
Cache Processing (seconds)	0.0 (+0.0)	0 (+0)	0 (+0)	0 (+0)	0.0 (+0.0)

Figure 10-48 Real Time Server Statistics: Absolute and Delta tab

Each column heading identifies what enabled function or associated server is being surveyed for statistical information. Each row identifies what statistical information is being retrieved. Statistical information displayed for each column includes:

- **Requests:** This field displays the number of requests to the enabled function or associated server identified at the top of the column.
- **Responses:** This field displays the number of responses sent by the enabled function or associated server identified at the top of the column.

- **Error responses:** This field displays the number of error responses sent by the enabled function or associated server identified at the top of the column. An error response example is the 404 “Page Not Found” response.
- **Non-cache responses:** This field displays the number of non-cached responses sent by the enabled function or associated server identified at the top of the column.
- **Cache responses:** This field displays the number of local memory cached responses sent by the enabled function or associated server identified at the top of the column.
- **Bytes received:** This field displays the number of bytes received by the enabled function or associated server identified at the top of the column.
- **Bytes sent:** This field displays the number of bytes sent by the enabled function or associated server identified at the top of the column.
- **Non-cache Processing (seconds):** This field displays the number of seconds of non-cached processing activity completed by the enabled function or associated server identified at the top of the column.
- **Cache Processing (seconds):** This field displays the number of seconds of cached processing activity completed by the enabled function or associated server identified at the top of the column.

The server Averages tab, shown in Figure 10-49 displays the average length of activity, in seconds, completed by the enabled function or associated server identified at the top of the column.

General	Absolute	Delta	Absolute and Delta	Averages
Average time per response:				
Type	Server handled	CGI	Tomcat	Customer module
Total (seconds)	0.00282	1.05132	0.34872	0
Non-cached (seconds)	0.00282	1.05132	0.34872	0
Cached (seconds)	0	0	0	0

Figure 10-49 Real Time Server Statistics: Average tab

Each column heading identifies what enabled function or associated server is being surveyed for statistical information. Each row identifies what statistical information is being retrieved. Statistical information displayed for each column includes:

- **Total (seconds):** This field displays the total time of activity completed by the enabled function or associated server identified at the top of the column.
- **Non-cached (seconds):** This field displays the average length of time of non-cached activity completed by the enabled function or associated server identified at the top of the column.
- **Cached (seconds):** This field displays the average length of time of cached activity completed by the enabled function or associated server identified at the top of the column.

10.9 References

Here are some other resources for you to read and learn more about how to improve the performance of your HTTP Server (powered by Apache) Web server:

- The iSeries Information Center is a good starting point for performance-related topics that include the logging of information with iSeries Collection Services:

<http://publib.boulder.ibm.com/iseries/v5r3/ic2924/info/rzahx/rzahxebushttp.htm>

- *iSeries Performance Capabilities Reference Version 5, Release 3*, SC41-0607, is the definitive guide to performance on the iSeries server. This manual, which is updated regularly, has a good chapter on Web serving and communications performance. You can find it on the Web at:

<http://www-1.ibm.com/servers/eserver/iseries/perfmgmt/resource.htm>

- *AS/400 HTTP Server Performance and Capacity Planning*, SG24-5645, is an IBM Redbook intended for iSeries programmers, network and system management professionals, and other information technologists that are responsible for designing, developing, and deploying Web-based applications and information systems. This IBM Redbook was written before the HTTP Server (powered by Apache) was brought to the iSeries server. Yet, it contains some useful advice on getting the best performance out of your Web application.
- *Performance Tools for iSeries*, SC41-5340, provides programmers with the information needed to collect data about the system, job, or program performance. It includes tips for printing and analyzing performance data to identify and correct inefficiencies that may exist. It also includes information about the manager and agent feature.



Getting started with Webserver Search Engine and Web Crawler

An interesting analogy may be OS/400. Users are given the ability to follow menus, search for commands, or directly enter commands. They can also prompt for additional context-sensitive information. All are designed for different types of users.

The key to any successful Web site is to offer a good clean way for people to navigate through the myriad of Web pages to find that one piece of information they need. The problem is that there are many different types of people that come to your Web site looking for information.

Some people like to follow a well thought out and defined hierarchy of information. You can use nested levels of navigation bars on the left, top, and right side of your Web pages to give a logical order to your site. You can also use site maps to give these type of people the big picture view of your site. Some like to follow the hypertext links found throughout the text and graphics that you have prepared. In a way, they read their way through to the information they are looking for. And some people simply like to search.

The webmaster's job is to provide all of these forms of navigation to your customers since you cannot control what kind of person they are. A good search engine, then, is part of the critical items you must add to your Web site.

On the iSeries server, the search engine comes in two logical pieces that are related to each other:

- ▶ iSeries Webserver Search Engine
 - Collects all documents into a single directory
 - Creates a search index
 - Creates a document list that contains a list of all the document paths
 - Customizes your search forms with supplied HTML section
 - Sets up your HTTP server correctly for the search forms
 - Keeps your index up to date
- ▶ iSeries Webserver Search Engine Web Crawler
 - Crawls the URL you provide and downloads the Web page
 - Builds a document list using downloaded Web pages
 - Create a search index using the document list

These two search engines are further explained in 11.1, "iSeries Webserver Search Engine" on page 308, and 11.2, "iSeries Webserver Search Engine Web Crawler" on page 309.

11.1 iSeries Webserver Search Engine

If you want to allow others to search through documents on your server, you need to set up your system to be a searchable site. Doing this is easy with the new iSeries Webserver Search Engine. There are just a few administrative tasks you need to do.

These tasks are summarized here:

1. Collect all of the related documents into a single directory on your iSeries server. You may use either the root (/) directory of the integrated file system (IFS) or the QSYS.LIB file system. Using the IFS system allows you to easily port your files from a PC onto the iSeries server.
2. Create a search index. An index is a collection of all of the selected documents in your directory. The documents are stored in a special indexed form. In the indexing process, the search engine takes each document provided in a document list and parses through it to create keys that are used in searches. The Webserver Search Engine uses very short character string keys. This indexed form allows for faster searching than can be done on documents that are not indexed.
3. The documents provided to the indexing function are contained in a document list that is automatically created when you create an index. A list can also be created through administrative forms or by hand.
4. After you create the search index, you can test it from the search administration form. This allows you to see all of the different options available to select for a search, such as fuzzy or precise.
5. Now you are ready to set up the Webserver Search Engine to run on your Web site. A short Hypertext Markup Language (HTML) section is supplied that you can add to your Web page. A Net.Data macro is also supplied that contains all of the HTML you need. This allows you to customize your search and search results forms. You may use the short HTML form supplying a few values if you are not comfortable using Net.Data. However, you must still copy the sample macro to your directory to make all of this work.
6. After you decide how you want to present your search forms, you need to make sure the HTTP server you use contains the correct directives in the configuration to run the Net.Data macro. You must also make sure that users can view the documents found on a search. A simple set of steps to do the necessary setup is provided for the HTTP Server (powered by Apache) and HTTP Server (original).
7. When all of this is completed, you are ready to perform searches.
8. You must keep your index up to date. If you modify your documents from time to time, make sure that your users can find the most current information. We supply a way for you to update your index. You can use the same document list you used when you originally created your index. We index any changed files that were previously indexed. You can also add a new set of documents to an index that already exists and delete some of the documents from your index. This is a matter of supplying different lists when you update the index.

For more information about how to use the iSeries Webserver Search Engine, see:

<http://www.ibm.com/servers/eserver/iseries/software/http/services/searchinfo.htm>

The online "How to" documentation is good for this feature. If you have not done so, we recommend that you consult the online documentation at this same Web site.

11.2 iSeries Webserver Search Engine Web Crawler

Web Crawler is a program that you can start from the same Search Setup forms that you use to set up your search engine. It works in much the same way as when you enter a Uniform Resource Locator (URL) on your browser and then click various links to go to new Web pages.

The crawling program starts by finding the URL that you provide. It downloads this Web page to your system and then continues to follow the links it finds. Each Web page that it links to is also downloaded until there are no more links to follow or your timer expires.

I Web Crawler was introduced to the HTTP Server (powered by Apache) with OS/400 V5R1.

Web Crawler extends the capability for building a document list. As each file is downloaded, the local path, plus the original URL, are added to your document list. You can then use this document list to create a search index. Search results for this type of index display the URL where the document was originally found rather than the local copy. When you find one of these documents in your search results, you are taken to the actual page that was found during crawling.

When you choose to build a document list by crawling Web sites, the session always runs as a background task whether it is initiated from the browser or one of the search CL commands. It takes several minutes to run at a minimum, depending, of course, on the maximum time you selected for the session to run, as well as other attributes you specified.

Web Crawler has some special features. It can go to any Web site, English or non-English, and process the downloaded files correctly for indexing and searching. If a site requires authentication, you can provide the necessary setup. Since Web Crawlers can run for quite a long time and consume a lot of your system storage, you can limit the time the crawler runs, the size of the files it can download, and the amount of storage it can consume. In addition, you can stop, pause, and resume your crawling session.

All of these features are on the Search Setup forms that are part of the HTTP Server Configuration and Administration.

For more information about how to use the iSeries Webserver Search Engine Web Crawler, see:

<http://www.ibm.com/servers/eserver/iseries/software/http/services/webcrawler.htm>



Apache Portable Runtime: Extending your core functionality

Consider these scenarios:

- ▶ What if you need to provide some kind of dynamic content to your Web site and it needs to be as efficient as possible?
- ▶ What if you want to perform some special logging that is outside what either your Web application or the HTTP Server (powered by Apache) does?
- ▶ What if you need to implement your own authentication of remote users for your Web application?
- ▶ What if you want to implement your own version of a server-side include (SSI) that would recognize a pattern of text in the outgoing Hypertext Markup Language (HTML) and replace it with some dynamic content?

The answer to all these “what if” questions is Apache Portable Runtime (APR). The APR is all about extending the core functionality of the Apache server in a manner that is portable between platforms. This chapter explores the APR Version 2.0 and writes a module for the iSeries server.

12.1 Apache module design overview

The design of the HTTP Server (powered by Apache) defines *modules*. Modules are operating system objects that can be dynamically linked and loaded to extend the nature of the HTTP Server (powered by Apache). Depending on the operating system, this is similar to:

- ▶ Windows Dynamic Link Libraries (DLL)
- ▶ UNIX shared object libraries
- ▶ OS/400 Integrate Language Environment (ILE) Service Programs

In this way, the Apache modules provide a way to extend a server's function. Functions commonly added by optional modules include:

- ▶ Authentication
- ▶ Encryption
- ▶ Application support
- ▶ Logging
- ▶ Support for different content types
- ▶ Diagnostics

A good example of a module that is shipped with your HTTP Server (powered by Apache) that extends the reach of the core Apache server is:

```
LoadModule ibm_ssl_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM
```

This service program is only loaded, linked, and used when you configure the LoadModule directive because you decided to encrypt your data using Secure Sockets Layer (SSL). The advantage of this is that the core Apache program can stay relatively small and tight until a particular function (as provided by a plug-in module) is needed. Then, with just a LoadModule directive and optionally some configuration directives, you can increase the functionality of your Web server with a corresponding increase in the working set size.

Apache core functions are those available in a standard Apache installation with no nonstandard modules. iSeries Apache Version 2.0 supports about 137 directives. Of those, 53 are in the core functions. The remainder are in separate modules that are compiled into the code. The LoadModule directive must be used to activate the directives in these modules.

As shown in Figure 12-1, the HTTP Server (powered by Apache)'s core functions are extended with a variety of different modules. In some cases (Common Gateway Interface (CGI) module and a Cookie module), the modules are built-in extensions to the server's core functions and do not need an explicit LoadModule to use. In other cases (for example, the SSL module), the modules are provided with the HTTP Server (powered by Apache) product on the iSeries but must be explicitly loaded with the LoadModule directive.

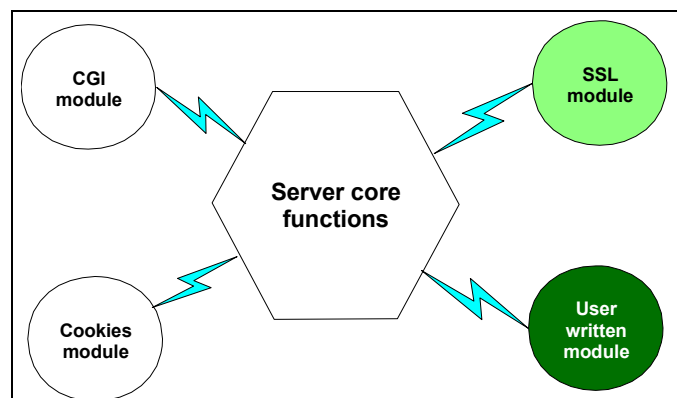


Figure 12-1 Modules expanding the functionality of the core Apache Web server

You can also write your own module to extend the core functionality of the HTTP Server (powered by Apache). This, in fact, is one of the biggest drawing points to the Apache Web server and a good example of why it is a popular HTTP server.

To understand how modules are plugged into the Apache server, you need a good understanding of *request handling*. Request handling is done in a series of steps called *phases*. The phases are:

- ▶ **Post read request:** Set of hooks called immediately after the incoming request is read.
- ▶ **URI to filename translation:** Hooks for mapping the incoming Uniform Resource Identifier (URI) to the physical file. Examples include aliasing, redirecting, rewriting, and so on.
- ▶ **Parse headers:** Hooks that need to read the incoming HTML headers and perform tasks based on what is in them.
- ▶ **Check access:** Hooks that determine whether the current context is restricted.
- ▶ **Authentication ID checking:** Hooks that verify who the user is.
- ▶ **Authentication access checking:** Hooks that verify if the user is authorized in the current context.
- ▶ **Type checker:** Hooks to determine the Multipurpose Internet Mail Extensions (MIME) type of the object requested.
- ▶ **Fixups:** Hooks that don't really fit anywhere else, such as ASCII/EBCDIC conversions or setting/resetting environment variables.
- ▶ **Insert filters:** Hooks for inserting input or output filters. A module either does this or registers the filter at startup time. If a module registers filters at startup time, the server adds those filters during this phase.
- ▶ **Sending a response back (handlers):** Hooks that possibly handle the request and respond to the client.
- ▶ **Logging:** Hooks to log any data to a log file that the module defines.

Tip: The example provided in 12.2, “Creating a module for the iSeries server” on page 315, is an example of a response handler (Sending a response back (handlers) in the list above). In this phase, *buckets and brigades* (see the next paragraph) are lined up and our code steps in to make a minor modification to the HTML right before it is sent back to the client.

For Version 2.0 of the Apache server, the Apache Software Foundation (ASF) provides the ability to modify data that was generated by an earlier module. This concept is called *buckets and brigades* as seen in Figure 12-2. The premise is that, after all is said and done, Web pages are nothing more than chunks of information:

- ▶ Each chunk is stored in a bucket.
- ▶ A list of buckets form a brigade.
- ▶ Lists of brigades can form a Web document.
- ▶ Filters operate on one brigade at a time.

The C language implementation of the structure shown in Figure 12-2 is a linked list of buckets.

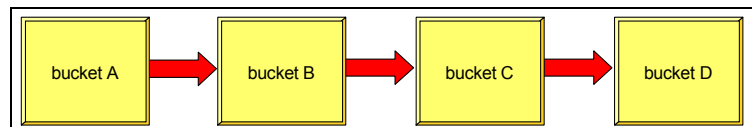


Figure 12-2 Many buckets in a row become a brigade

As you now know, the iSeries has integrated the 2.0 version of the Apache server with the IBM HTTP Server for iSeries. Much of the rest of the world, however, is still back at version 1.3 of the Apache server. A big difference between version 1.3 and 2.0 of the Apache server is that the APR is new for version 2.0. To bring a module written to version 1.3 to the iSeries server, you should first update it to the new version 2.0 APR module. Then, the port to iSeries should be fairly easy.

The APR found with version 2.0 of the Apache server is actually independent of the Apache HTTP 2.0 server. Technically, APR is a separate Apache product altogether and can exist alone. Users of APR can create their own applications using APR and not touch the Apache HTTP 2.0 server.

12.1.1 Documentation and resources

For you to successfully implement a more complex module using the APR of Apache, you need to study the following documentation:

- ▶ V5R3 HTTP Server Documentation Center:
Select the document depending on the version you are using, V5R3, V5R2, or V5R1. Inside the V5R3 IBM HTTP Server for iSeries Documentation Center, click **Programming → HTTP Server (powered by Apache) and Apache portable runtime application programming interfaces → Compile and configure modules on HTTP Server (powered by Apache)**. You can find the documentation center at:
<http://www.ibm.com/servers/eserver/iseries/software/http/docs/doc.htm>
- ▶ Apache Software Foundation home page:
<http://www.apache.org>
- ▶ Apache Software Foundation documentation:
<http://httpd.apache.org/>
Select **Apache 2.0** under Documentation.
- ▶ Links to many Apache Software Foundation resources on APR:
<http://apr.apache.org/>
- ▶ Apache modules registry:
<http://modules.apache.org/>

These online magazines often provide Apache 2.0 leading edge advice and support, especially in the area of writing modules:

- ▶ *ApacheToday* news and information online:
<http://www.apachetoday.com>
- ▶ *Apache Week* news and information online:
<http://www.apacheweek.com>
- ▶ *Onlamp* news and information online:
<http://www.onlamp.com/apache/>

12.2 Creating a module for the iSeries server

The best way to learn how to create a module for the iSeries server is to just do it.

12.2.1 The task at hand

Our goal is to create a module that adds text (most likely HTML) to the start of a Web page. We do so only within the context in which we define an output filter handler. That is, we use the same way that all Apache directives can inherit or override the settings of directives found above this context. This, in effect, provides private storage for our module. There is one per Directory context.

In our example, Table 12-1 defines many of the features of our module.

Table 12-1 Definition of the HeaderFilter module

Feature	How is it defined or used
DocumentRoot	/tcp52d00/basicconfig/itsoco
HeaderFilter active in context	<Directory /tcp52d00/basicconfig/itsoco/people>
Define the text that will be added just before the start of the <html> tag with all pages sent from within the active context.	HeaderText "<center><i>Listen to all the People</i></center>"
Cause the header_module module to be loaded	LoadModule header_module /QSYS.LIB/TCP52L00.LIB/MOD_HEADER.SRVPGM

12.2.2 Source code and comments

This section provides the C language source code and annotated comments to help guide your understanding of the logic behind the code.

mod_header.c source code and annotation

Example 12-1 shows the C language source code of the module mod_header.c. The numbered annotations found in the source code are described in "Comments to mod_header.c" on page 318.

You can find this source code in:

- **Library:** TCP52L00
- **Source file:** QCSRC
- **Member:** MOD_HEADER

Example 12-1 C language source code for the mod_header.c module

```
/* 1 */
#include "apr_strings.h"
#include "apr_xlate.h"
#include "ap_config.h"
#include "util_filter.h"
#include "httpd.h"
#include "http_config.h"
#include "http_request.h"
#include "http_core.h"
#include "http_protocol.h"
#include "http_log.h"
#include "http_main.h"
#include "util_script.h"
#include "http_core.h"
```

```

#include "util_charset.h"

module AP_MODULE_DECLARE_DATA header_module; /* 2 */
/* 3 */
typedef struct header_rec {
    const char *headert;
} header_rec;

/*
 * 4 Handle the HeaderText directive
 */
static const char *add_header_text(cmd_parms *cmd, void *dummy,
                                   const char *arg)
{
    header_rec *d = dummy;

    /* store the text for the header. */
    d->headert = apr_pstrdup(cmd->pool, arg);
    return NULL;
}

/*
 * 5 Define the directives
 */
static const command_rec dir_cmds[] =
{
    AP_INIT_TAKE1("HeaderText", add_header_text, NULL,
                  ACCESS_CONF || OR_FILEINFO,
                  "text for a header"),
    {NULL}
};

/*
 * 6 Create the module specific structure
 */
static void *create_header_config(apr_pool_t *p, char *dummy)
{
    header_rec *new =
        (header_rec *) apr_pcalloc(p, sizeof(header_rec));

    new->headert = NULL;
    return (void *) new;
}

/* 7 */
typedef struct header_struct {
    int state;
} header_struct;

/*
 * 8 Define the filter to add the header text to the request.
 */
static int header_filter(ap_filter_t *f, apr_bucket_brigade *bb)
{
    header_struct *ctx = f->ctx;
    header_rec *conf;
    apr_bucket *e;

    /* get the module specific structure for the current context. */
    conf = (header_rec *)ap_get_module_config(

```

```

        f->r->per_dir_config,
        &header_module);

/* If the current context has not been created, create one */
if (ctx == NULL) {
    f->ctx = ctx = apr_palloc(f->r->pool, sizeof(*ctx));
}

/*
 * If the context state is 0 (meaning we haven't yet done this)
 * AND if the current object being processed is text/html
 * Then we will process what was defined in the HeaderText directive
 */
if((ctx->state == 0) &&
    (!strcasecmp(ap_field_noparam(f->r->pool, f->r->content_type),
        "text/html"))) {
    ctx->state = 1; /* Indicate that we've been here */
    /* If HeaderText directive has been defined for the current context...*/
    if (conf->headert) {
        char *headertext = apr_palloc(f->r->pool,
            strlen(conf->headert));

        apr_size_t in_length = strlen(conf->headert);
        apr_size_t out_length = in_length;
        /* We must convert the text string from EBCDIC to ASCII.
         * ap_locale_to_ascii is defined by the server at startup time.
         */
        apr_xlate_conv_buffer(ap_locale_to_ascii,
            conf->headert, &in_length,
            headertext, &out_length);

        /* Create the bucket to store the ASCII text */
        e = apr_bucket_immortal_create(headertext,
            strlen(conf->headert));

        /* Insert the bucket at the beginning. */
        APR_BRIGADE_INSERT_HEAD(bb, e);
    }
}

/* Pass the brigade on... */
ap_pass_brigade(f->next, bb);
return APR_SUCCESS;
}

/*
 * 9 Register any hooks or filters needed for this module
 */
static void header_register_hook(apr_pool_t *p)
{
    ap_register_output_filter("HEADERFILTER",
        header_filter,
        NULL,
        AP_FTYPE_CONTENT);
}

/* 2 */
module AP_MODULE_DECLARE_DATA header_module = {
    STANDARD20_MODULE_STUFF,
    create_header_config, /* create per-directory config structure */
    NULL,                /* merge per-directory config structures */
    NULL,                /* create per-server config structure */
    NULL,                /* merge per-server config structures */
}

```

```

    dir_cmds,          /* command apr_table_t */
    header_register_hook /* register hooks */
};

```

Comments to mod_header.c

Here are the comments to the code presented in the previous section. Each numbers corresponds to the bold numbers in Example 12-1 on page 315:

1. The header files included in this example are only to define all application programming interfaces (APIs) used in the example. These and a lot more header files are located in your iSeries server's integrated file system (IFS) directory /QIBM/ProdData/HTTP/include/.
2. This is the *Command Module Structure*, which is well known to the server. It contains:
 - Standard Apache 2.0 module items
 - Function pointer to create per-directory configuration structure
 - Function pointer to merge per-directory configuration structures
 - Function pointer to create per-server configuration structure
 - Function pointer to merge per-server configuration structures
 - Pointer to command table
 - Function pointer to register hooks function

This module also needs to be exported from the service program that you create. See 12.2.3, "Compiling, linking, and exporting your service program" on page 319.

3. Here we declare some module specific storage that will later be used to store the header text that will come from the configuration file's HeaderText `<center><i>Listen to all the People</i></center>` directive.
4. This is the add_header_text function. It is one of the parameters used in the command table.
5. The command table and command handler define:
 - The server directives
 - Functions to handle those directives

The command table is composed of:

- Directive name
- Configuration action routine. In our case, this is the add_header_file function.
- Additional argument to include in call
- Where directive is valid
- Directive description

The APR module support at 2.0 defines a series of directive initializers used to define how many arguments (parameters) are passed.

In our case, we use AP_INIT_TAKE1. Since we defined the directive to take one argument (AP_INIT_TAKE1), the string following the directive in the configuration file needs to be quoted like this:

```
HeaderText "<center><B><i>Listen to all thePeople</i></B></center>"
```

See 12.2.4, "Activating via configuration" on page 320, for the details.

The list of directive initializers include:

- **AP_INIT_RAW_ARGS**: Function parses the command line itself
- **AP_INIT_TAKE1**: One argument
- **AP_INIT_TAKE2**: Two arguments
- **AP_INIT_ITERATE**: One argument, occurring multiple times
- **AP_INIT_ITERATE2**: Two arguments; the second occurs multiple times

- **AP_INIT_FLAG:** Values “On” or “Off”
 - **AP_INIT_NO_ARGS:** No arguments
 - **AP_INIT_TAKE12:** One or two arguments
 - **AP_INIT_TAKE3:** Three arguments
 - **AP_INIT_TAKE23:** Two or three arguments
6. Create a per-directory configuration. This, in effect, provides private storage for our module. One per Directory context. This can be merged with previous context in the tree in the same way that all Apache directives can inherent or override the settings of directives found above this context.
 7. Define the filter context structure. This structure is designed to keep track of things needed in case this filter is called multiple times in a context.
 8. This is the main declaration and code that is our output filter.
 9. Register the output filter hook. All register hooks are called at startup time. They are designed for modules to register the filters (as in this example) and to specify which phases the module wants to hook. This can be done by calling APIs for specific hooks (for example calling `ap_hook_translate_name` to hook the URI to filename translation phase).

12.2.3 Compiling, linking, and exporting your service program

After you create your module, it is time to compile and then create and export the `header_module` service program.

Note: A change to the HTTP Server (powered by Apache) in V5R2 requires a new parameter `TERASPACE(*YES)` on the Create C Module (`CRTCMOD`) command. If you recompile your programs with this option and then rebuild your service program, the performance of your service program should improve.

Compiling the service program

For V5R2 and V5R3, you enter the following command (all as one command):

```
CRTCMOD MODULE(TCP52L00/MOD_HEADER)
SRCSTMF('/QSYS.LIB/TCP52L00.LIB/QCSRC.FILE/MOD_HEADER.MBR') DEFINE(AS400)
LOCALETYPE(*LOCALE) TERASPACE(*YES) INCDIR('/qibm/proddata/httpa/include')
```

For V5R1, you enter the following command (all as one command):

```
CRTCMOD MODULE(TCP52L00/MOD_HEADER)
SRCSTMF('/QSYS.LIB/TCP52L00.LIB/QCSRC.FILE/MOD_HEADER.MBR') DEFINE(AS400)
LOCALETYPE(*LOCALE) INCDIR('/qibm/proddata/httpa/include')
```

For V4R5, you enter the following commands:

```
CHGCURDIR DIR('/qibm/proddata/httpa/include')
```

```
CRTCMOD MODULE(TCP52L00/MOD_HEADER)
SRCSTMF('/QSYS.LIB/TCP52L00.LIB/QCSRC.FILE/MOD_HEADER.MBR') DEFINE(AS400 '_MULTI_THREADED')
LOCALETYPE(*LOCALE)
```

Creating and exporting the service program

For either V5R3, V5R2, V5R1, or V4R5, you create a service program export source member. You can find this source code in:

- **Library:** TCP52L00
- **Source file:** QSRVSR
- **Member:** MOD_HEADER

It contains:

```
STRPGMEXP PGMLVL(*CURRENT)
  EXPORT SYMBOL("header_module")
ENDPGMEXP
```

Then create the mod_header service program:

```
CRTSRVPGM SRVPGM(TCP52L00/MOD_HEADER) MODULE(TCP52L00/MOD_HEADER) EXPORT(*SRCFILE)
SRCFILE(TCP52L00/QSRVSR) SRCMBR(MOD_HEADER) BNDSRVPGM(QHTTPSVR/QZSRAPR QHTTPSVR/QZSRCORE
QHTTPSVR/QZSRXMLP QHTTPSVR/QZSRSDBM)
```

Note: If you create a module with *PUBLIC *RWX authority, there should be no authority considerations. If *PUBLIC is *EXCLUDE, authority has to be granted explicitly to the profile that the server instance will run under as well as user profile QTMHHTTP.

12.2.4 Activating via configuration

Then add these directives to your configuration file:

- ▶ To cause the module header_module to be loaded by the HTTP Server (powered by Apache) at server startup time:
`LoadModule header_module /QSYS.LIB/TCP52L00.LIB/MOD_HEADER.SRVPGM`
- ▶ Within the context in which you want the module header_module to be executed, use HeaderText as a directive to define the HTML text that will be added to the start of any HTML page within this context. Consider this example:

```
<Directory /tcp52d00/basicconfig/itsoco/people>
HeaderText "<center><B><i>Listen to all the People</i></B></center>"
</Directory>
```

Attention: When using the Display Configuration File option of the IBM Web Administration for iSeries interface, the administration program checks for syntax errors. A directive with an incorrect syntax is displayed in red. Directives that are added via user modules are also displayed in red even if they are correct.

12.2.5 Testing header_module

To test header_module, save your configuration file and then start your server:

- ▶ **Server:** PBABASIC00
- ▶ **Listen:** port 8000
- ▶ **DocumentRoot:** /tcp52d00/basicconfig/itsoco

With the Uniform Resource Locator (URL) <http://as20:8000/index.html>, you should see your normal home page unchanged as shown in Figure 12-3.



Figure 12-3 *header_module* is not evoked for this context; no changes to HTML

Next, using the navigation bar on the left, click **People**. The pages to support the People section of our small Web application are defined within the context in which *header_module* is registered. This causes the People page to look like the example in Figure 12-4.



Figure 12-4 *header_module* evoked for this context; notice 'Listen to all the People'

Further, if you view the source for the page, you see:

```
<center><B><i>Listen to all the People</i></B></center>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
<META NAME="Generator" CONTENT="NetObjects Fusion 5.0.1 for Windows">
<TITLE>People</TITLE>
</HEAD>
<BODY ...
```

12.2.6 Debugging

Here are two methods to debug your module:

- Compile your module with debug views turned on, for example:

```
CRTCMOD MODULE(TCP52L00/MOD_HEADER) DBGVIEW(*ALL)  
SRCSTMF('/QSYS.LIB/TCP52L00.LIB/QCSRC.FILE/MOD_HEADER.MBR') DEFINE(AS400)  
LOCALETYPE(*LOCALE) TERASPACE(*YES) INCDIR('/qibm/proddata/httpa/include')
```

You can then use the Start Service Job (STRSRVJOB) and Start Debug (STRDBG) commands to set breakpoints in your module.

- You can directly add trace points into your code. These trace points can be turned on and off with the Trace TCP/IP Application (TRCTCPAPP) CL command.

The APIs for this are defined in the API guide. For more information about this, click the Reference documentation for HTTP Server link on the iSeries Information Center at:

<http://publib.boulder.ibm.com/infocenter/iseries/v5r3/ic2924/index.htm>

Using AP_ERROR_TRACE, AP_INFO_TRACE, or AP_VERBOSE_TRACE at various places in the code accomplishes this. Then the user only has to use the TRCTCPAPP command to turn on and off the trace for the instance. The trace points that were added appear in the TRCTCPAPP output depending on the level coded versus the level requested.



Problem determination: When things do not go as planned

Setting up and tuning an HTTP server requires time, testing, and patience. This chapter helps you face the problems that you may encounter in both phases. It gives you a better understanding of the practices and tools to assist you in your work.

13.1 The art of problem determination

This section is by no means a ready solution for all of your Web serving woes. It is only intended as a quick guide to help solve the most common problems you will encounter when configuring and managing your HTTP Server (powered by Apache).

If you are experiencing a special problem, skip Table 13-1. Instead read 13.2, “Tools of the trade” on page 327, which explains the detailed problem determination tools and techniques. Use Table 13-1 as a quick checklist and a guide during problem determination.

Note: If the graphical user interface (GUI) is not doing what you want it to do, then see Table 13-2.

Table 13-1 Problem determination checklist

Symptom	What to do
The server does not start or does not stay active.	<ul style="list-style-type: none">▶ Manually start your server from a green screen using the Start TCP/IP Server (STRTCPSVR) command. Look for messages in your job log. The completion message CPC1221 informs you that job NNNNNN/QTMHHTTP/SERVERNAME was submitted. You can use this data in the Work with Job (WRKJOB) command to retrieve the server job log. Refer to 13.2.1, “Working with configuration files” on page 327, for detailed information about server job logs.▶ Make sure that the user profile QTMHHTTP or the profile you chose as the default (see Figure 13-5 on page 330) fits this profile:<ul style="list-style-type: none">– Exists on the server– Is enabled– Has no password expiration date set▶ If using Net.Data or Common Gateway Interface (CGI) programs, repeat the previous step for user profile QTMHHTTP1.▶ Check that all software requirements are met. Refer to Chapter 2, “From zero to powered by Apache” on page 17, for additional information.
We can't find the “myserver” message using Internet Explorer.	<ul style="list-style-type: none">▶ Verify that the HTTP server is active. Also PING the server using both the name and Internet Protocol (IP) address. If name fails and IP succeeds, this is a Domain Name System (DNS) problem. Either add the server name and IP to your client's hosts table or contact your DNS server administrator.
There was no response. The server must be down or is not responding to the message using Netscape.	<ul style="list-style-type: none">▶ Make sure that you enter the Uniform Resource Locator (URL) in your address bar exactly like the following examples if you are using Secure Sockets Layer (SSL) or Transport Layer Security (TLS): <code>http://servername:port</code> <code>https://servername:secureport</code>▶ Check your browser's proxy setting and set it appropriately. If your Web client is connected directly to the HTTP Server (powered by Apache), then try deleting the client proxy configuration to see if that makes a difference.▶ Use the Work with Subsystem Jobs (WRKSBSJOB) command to see whether the QHTTSPSVR subsystem is hosting your HTTP server jobs. If you are using Web application servers that run on top of the HTTP server, such as WebSphere or Tomcat, use the Work with Active Jobs (WRKACTJOB) command and set the JOB parameter to your server name instead.

Symptom	What to do
<i>Continued</i>	<ul style="list-style-type: none"> – If you don't see any job named after your server, refer to the previous tip "The server does not start". – If the jobs are active, use the Work with TCP/IP Network Status (NETSTAT) command with the *CNN option. Look at the port numbers listed in the Local Port column. You should be able to find port 80 (or the port you chose for your server). – If your port is not listed, look for clues in the server job logs. See 13.2.2, "Job logs" on page 329, for details. <p>Note: When you are unable to bind to a specific port, the HTTP server jobs remain active for approximately five minutes. During this time, the bind operation is attempted every five seconds, and the message HTP803D is posted to the job log. Should this occur, make sure that no other application is already using the same port.</p>
HTTP error message 404 - File not found.	<ul style="list-style-type: none"> ► Make sure that the file you requested meets this criteria: <ul style="list-style-type: none"> – Exists in your document root or in your current path. – Can at least be read by your server user profile. Use the Display Authority (DSPAUT) command for this purpose. See Figure 13-5 on page 330 if you don't know which user profile is running the server. ► Look at the access_log and error_log files. You'll find more information about log files in 13.2.3, "Server logs" on page 331.
CGI program or Net.Data macro is not running.	<ul style="list-style-type: none"> ► Use the Display Authority (DSPAUT) command to verify that the program can be run by user QTMHHTTP1. ► Check the server job log for obvious clues. ► Refer to <i>HTTP Server for iSeries Programming</i>, GC41-5435, for CGI program debugging tips. See 13.2.4, "Net.Data logs and traces" on page 340, for information about Net.Data debug procedures, or look at a sample Net.Data configuration in 7.3, "Net.Data: A ready-made scripting tool" on page 161.
Unsatisfying performance.	<ul style="list-style-type: none"> ► See 10.1, "iSeries Web server performance components" on page 226.
LDAP authentication does not work	<p>When Lightweight Directory Access Protocol (LDAP) user authentication fails for HTTP basic authentication, the error_log of your HTTP server instance is the starting point for debugging. There can be multiple reasons why LDAP authentication fails. The following examples give some hints on where to look for possible causes.</p> <ul style="list-style-type: none"> ► Error message in error_log: <pre>[Fri Oct 01 13:35:18 2004] [error] ZSRV_MSG0080: Unable to authenticate HTTP server for realm 'FRA822 LDAP Server': Error is Invalid credentials.</pre> <p>When using LDAP authentication, the HTTP server is an LDAP client that needs to bind to the LDAP server. The LDAP configuration file (see 6.2.3, "Authentication by LDAP entries" on page 113) contains the bind distinguished name (DN) and the password that is used by the HTTP server to authenticate to the LDAP server. When receiving the previous message, the HTTP server was not able to authenticate to the LDAP server. To correct the problem, check the administrator DN and password.</p> ► Error message in error log: <pre>[Fri Oct 01 13:32:53 2004] [error] ZSRV_MSG0066: Unable to find entry with search filter '(&(objectclass=person)(!(cn=barle*)(uid=barle)))': Returning 401 error</pre> <p>This message indicates that the HTTP server successfully bound to the LDAP server and used the search filter '(&(objectclass=person)(!(cn=barle*)(uid=barle)))' to look for an entry where the objectclass is person and the common name (cn) attribute or the uid attribute contains the value barle. In this case, the user does not exist. However, the message is also issued when the search filter contains errors. To correct the problem, verify that the user as displayed in the search filter part of the log does exist.</p>

Symptom	What to do
LDAP authentication does not work (continued)	<p>You can also verify the search filter definition in the LDAP properties file you created during the LDAP authentication setup.</p> <ul style="list-style-type: none"> ▶ Error message in error_log: <pre>[Fri Oct 01 13:49:16 2004] [warn] ZSRV_MSG0063: Basic authentication failure for user 'cn=Thomas Barlen,o=company00': Error is Invalid credentials</pre> <p>When you receive an error such as this, the HTTP server successfully connected to the LDAP server and looked up an entry with the specified search filter. Also the LDAP server returned the hashed password to the HTTP server. You can easily recognize this by the DN of the entry in the log file. However, this time the password is the problem. To correct the problem, the user must enter the correct password or the administrator needs to reset the password for the user.</p>

Table 13-2 identifies some common problems with the Administration GUI used to configure and manage your HTTP Server (powered by Apache).

Table 13-2 Common GUI problems

A common problem	What you can do
msgCEE0200 is in the ADMIN job log.	Verify that JDK 1.3 (5722-JV1 option 5) is installed on your system. See Chapter 2, "From zero to powered by Apache" on page 17, for a list of pre-requisites.
Password prompt appears several times.	Check your browser security setting. Both JavaScript and Cookies must be enabled.
Frames do not display properly.	
Buttons do not work.	
The page is too large to fit in the browser window.	<p>Most GUI pages are best viewed at a 1024x768 screen resolution. Here are other tips for adjusting the view:</p> <ul style="list-style-type: none"> ▶ The frames you see on the screen can be resized. Click a frame border and drag it to a different position. This new layout is maintained for the whole session. ▶ Try a smaller font size. Select View → Text Size from the menu options in Internet Explorer, View → Decrease Font in Netscape Navigator, or Document Zoom in the Opera list.
ADMIN does not start.	Use the Work with Job (WRKJOB) CL command to retrieve the server job log, and look for significant clues.
ZUI_50004: OS/400 user profile USERNAME does not have *IOSYSCFG authority, which is required to use the configuration and administration interface of IBM HTTP Server for iSeries.	Remember that you are ultimately dealing with OS/400 configuration files. OS/400 requires *IOSYSCFG authority from any user attempting to alter configuration files. Sign on with a more powerful profile.
Other interesting tips.	DSPAUT OBJ('/') should not show *EXCLUDE for *PUBLIC.
	Use the Work with Object Links (WRKLNK) CL command to browse the IFS path /QIBM/UserData/HTTPPA/admin/logs where the ADMIN error logs are stored. For additional information, see the following section.

13.2 Tools of the trade

Let us now look at the tools that make our everyday tasks easier. We identify what they are and how we can make the most out of them. This section takes you deeper into problem determination and shows how to work with configuration files, logs, and iSeries native instruments such as job logs and application traces.

13.2.1 Working with configuration files

Manually editing the configuration file requires care, patience, knowledge of the configuration directives, and a good backup of the original file. We recommend that you do not manually edit your `httpd.conf` file unless you really know what you are doing and you have solid experience with the Apache configuration directives.

The recommended way to change or create your HTTP Server (powered by Apache) configuration is to use the GUI. The GUI also supports good tools for displaying and editing configuration files.

From the main Configuration panel, select the **Display Configuration File** option. This opens the content of your configuration file just as the server sees it. This is really important if you manually altered the configuration file using the green screen Edit File (EDTF) utility. EDTF is a quick and handy tool for editing files on the iSeries server, but it doesn't provide the additional error highlighting that the GUI has.

Note: This check on the configuration file is similar to what the Apache native `-t` switch does. See 13.2.7, "Other startup parameters" on page 351, for `-t` and other switches.

For an example, look at the second to last line in Figure 13-1, where we purposely added the unsupported directive `AddModule` to our `httpd.conf` file.

```
Edit file: /www/itsonew/conf/httpd.conf
Record :      1  of      28 by 18          Column :      1   101 by 131
Control :

.....1.....2.....3.....4.....5.....6.....7.....8.....
*****Beginning of data*****
# Configuration originally created by Create HTTP Server wizard on Wed Sep 29 15:03:49 C
Listen *:8022
DocumentRoot /www/itsonew/htdocs
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogMaint logs/error_log 7 0
AddModule mod_cgi
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
```

Figure 13-1 The EDTF utility to edit the `httpd.conf` file

We save the file, and then go back to the GUI main configuration panel (Figure 13-2). Locate the **Tools** section at the bottom of the display and click **Display Configuration File**.

The latest selection displays the content of the httpd.conf file as shown in Figure 13-2. As you can see, the unsupported directive is highlighted and a comment is placed underneath. Each line is also numbered, making errors easier to spot. Also notice that the directives used by default are crossed out and are destined to be deleted.

Note: Since directives contain default values, which is what the server uses if it does not find them, there is no reason to have them in the configuration. To limit the configuration file size and improve its readability, they are deleted. This occurs when you are in the configuration GUI for the specific directive and click OK or Apply.

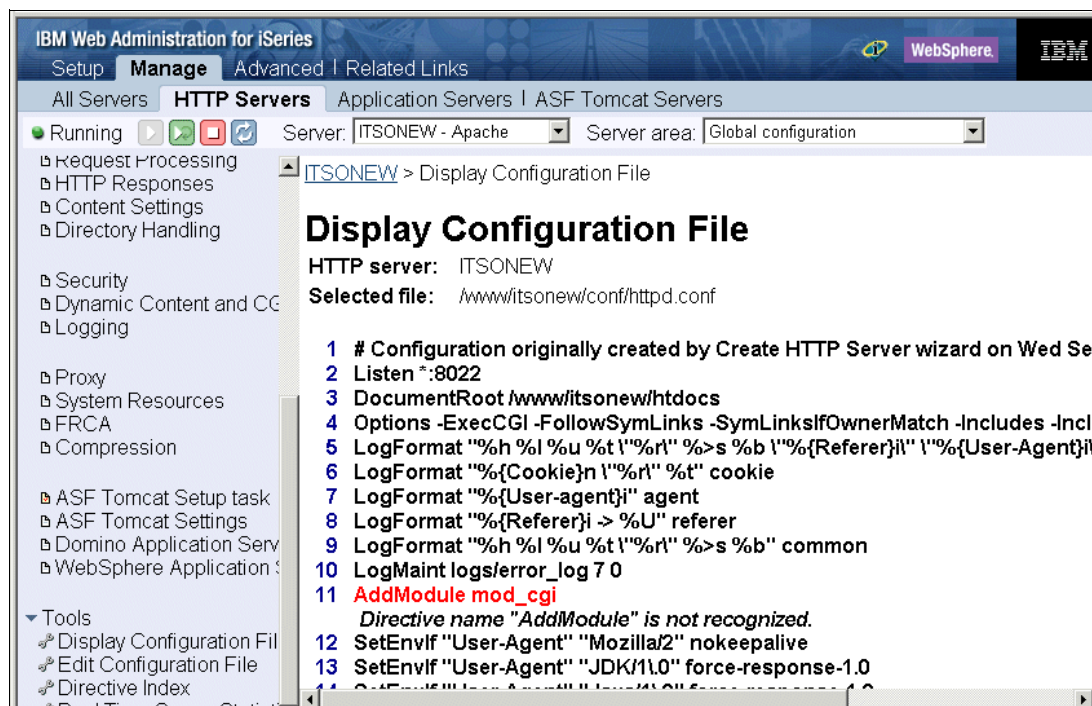


Figure 13-2 Display Configuration File: Line 10 indicates a problem with the configuration file

Let's see what happens if we try to start the server using this configuration file. The server ends immediately after parsing the invalid AddModule directive in line 10. The HTP8006 and HTP8008 error messages in the job log (as shown in Figure 13-3) clearly indicate where the problem lies. Now refer back to Figure 13-2 to see that line 10 of our configuration file is indeed the invalid AddModule directive.

Note: Not all directives that are marked as not recognized are actually incorrect directives. The GUI checks for directives that it knows. You could still see a directive in error for directives that belong to modules you have written or modules that are not part of the IBM HTTP Server for iSeries product. However, the joblog and the GUI are an excellent starting point for problem determination.

For more information about collecting, locating, and analyzing job logs, continue with the following section.


```

                                Display Spooled File
File . . . . . : QPJOBLOG                                         Page/Line 1/28
Control . . . . . Columns 1 - 130
Find . . . . .
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...0...+...1...+...2...+
HTP8006   Diagnostic      40  05/01/03 13:21:51.124792 QZSRAPR  QHTTSPVR  *STMT  QZSRCORE  QHTTSPVR  *STMT
      From module . . . . . : QZSRSDNM
      From procedure . . . . . : sendMessageToJobLog
      Statement . . . . . : 11
      To module . . . . . : HTTP_CONF1
      To procedure . . . . . : ap_walk_config_sub
      Statement . . . . . : 11
      Message . . . . . : Directive not recognized.
      Cause . . . . . : Directive AddModule is not a recognized HTTP server
                        directive. The HTTP server did not start. Recovery . . . : Correct or
                        remove the directive. Then start the HTTP server again. Technical
                        description . . . . . : See the HTTP server documentation on
                        configuration and administration for more information.
HTP8008   Escape      40  05/01/03 13:21:51.126128 QZSRAPR  QHTTSPVR  *STMT  QZHBHTTP  QHTTSPVR  *STMT
      From module . . . . . : QZSRSDNM
      From procedure . . . . . : sendEscapeWithMessageFile
      Statement . . . . . : 4
      To module . . . . . : HTDAEMON
      To procedure . . . . . : BigSwitch_FiPPc
      Statement . . . . . : 1070
      Message . . . . . : HTTP Server Instance ITSONEW failed during start-up.
      Cause . . . . . : HTTP Server instance ITSONEW failed because of a
                        configuration error on line 10 in configuration file
                        /www/itsonew/conf/httpd.conf. Note: If the specified directive is either a
                        container directive (e.g. <Directory>), or a directive within a container,
                        the line number identified above may not be correct. In that case, you will
                        need to verify that all directives in the container, and the container
                        itself do not have configuration errors. Recovery . . . : See previous
                        job log messages. Correct the problem and start the server again.

```

Figure 13-3 Display spooled file: HTP8006 and HTP8008

13.2.2 Job logs

HTTP server job logs are the first place to look for information whenever an abnormal ending occurs. Their content can be more or less detailed, depending on the message logging settings in the job description (JOB) in use. The JOB used by the HTTP Server (powered by Apache) is QZHBHTTP in the QHTTSPVR library. Changing its message logging settings always influences the content of your server job logs. Figure 13-4 shows the default values for this IBM-supplied JOB.

Changing the Text setting from *NOLIST to *MSG or *SECLVL can be extremely useful for debugging purposes. See the online help for the Change Job Description (CHGJOB) CL command for usage information. Also remember that *SECLVL generates a highly verbose job log for every server job. Therefore, do *not* choose it as a default setting.

Display Job Description		System: ASM20
Job description:	QZHBHTTP	Library: QHTTSPVR
Message logging:		
Level	:	4
Severity	:	0
Text	:	*NOLIST
Log CL program commands	:	*NO
Accounting code	:	*USRPRF
Print text	:	*SYSVAL
Routing data	:	HTTPWWW
Request data	:	*NONE
Device recovery action	:	*SYSVAL
		More...
Press Enter to continue.		
F3=Exit F12=Cancel		

Figure 13-4 Display Job Description: Default message logging

Job logs are always produced under the default QTMHHTTP profile unless you choose to use a different one, adding a ServerUserID directive in your configuration file. Figure 13-5 and the following steps show how to set a default user using the GUI:

1. In the left pane, under Server Properties, click **General Server Configuration**.
2. In the General Server Configuration panel, click the **Advanced** tab.
3. In the Server user profile field, type the user profile name that is used by the HTTP server. If no user profile is specified, it defaults to QTMHHTTP.

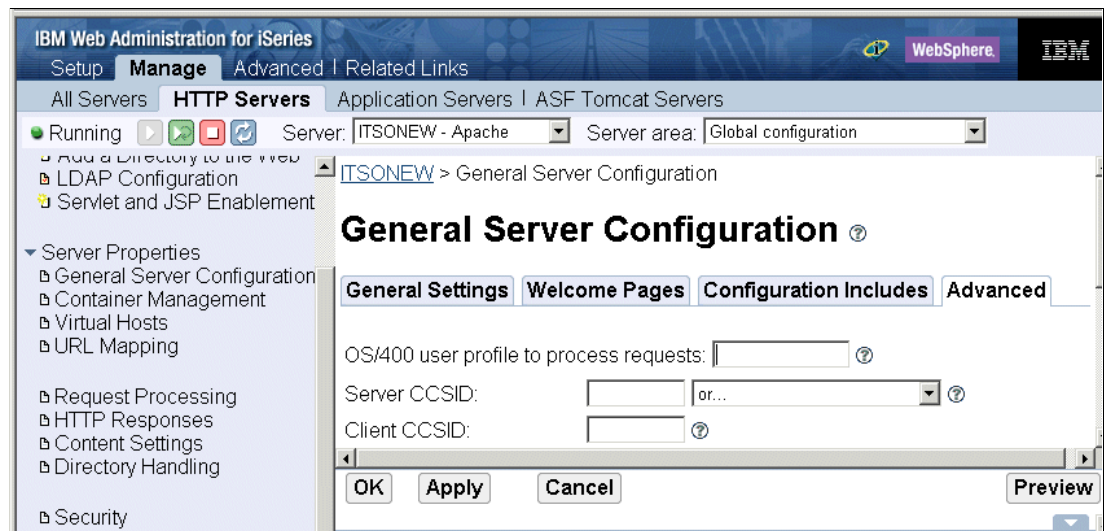


Figure 13-5 General Server Configuration: Server user profile

Messages in your server job logs often contain helpful hints for problem determination, for example:

- ▶ The name of a failing module
- ▶ Illegal configuration options
- ▶ Usage of a deprecated directive
- ▶ The number of the line where an error was found

To determine where the problem lies, you can also look up the line number referred to in the message body with the Display Configuration File menu option as explained in 13.2.1, “Working with configuration files” on page 327.

13.2.3 Server logs

Server logs are most useful for monitoring server activity and keeping track of user access, as well as being a valid aid in debugging. By carefully examining these logs, you can discover the reason behind the most common error messages and eventually point out configuration errors. Figure 13-6 shows the logging options.

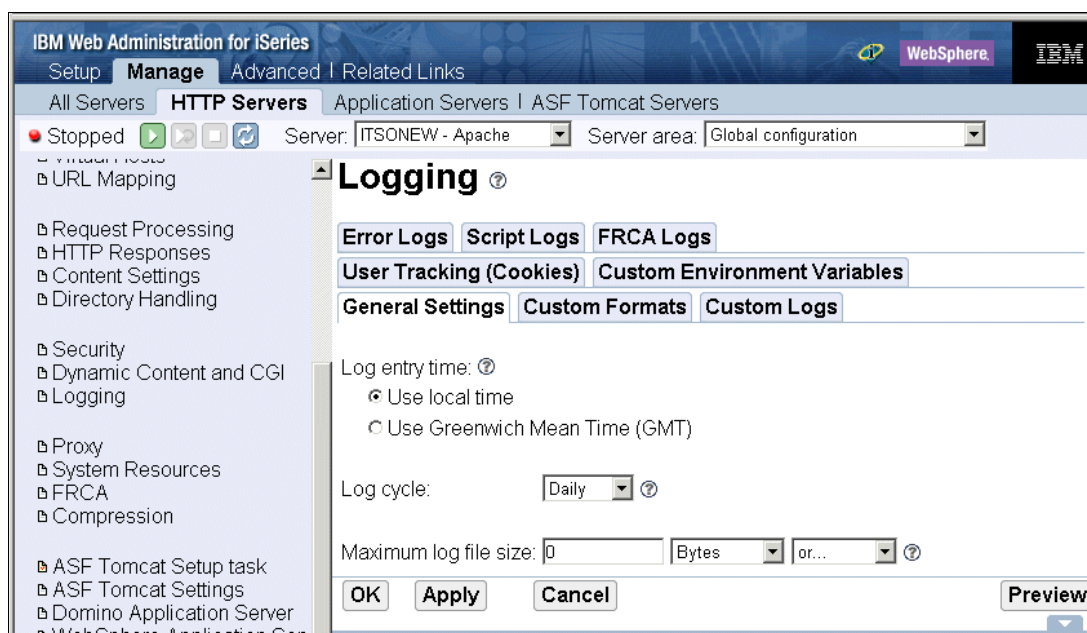


Figure 13-6 Logging server activity: General Settings page

General settings

The General Settings page allows you to configure settings that apply to all server log files such as selecting which time format each log entry time stamp will follow, controlling how often log files are closed and new log files created, and limiting the size of any defined log file. This is also a security mechanism, to protect the server in case of a denial of service attack from filling up direct access storage device (DASD) by producing huge log files.

The Log cycle parameter controls how often log files are closed and new log files created. Maximum log file size limits the size of any defined log file on the system.

Custom logs

The Custom Logs page allows you to configure various log attributes, such as the format for the information in the log file, rules for excluding entries from the log file, and client side information logging. Each server configuration file contains information about the type of log files the server will create. Logging information allows you to track and generate reports on your server's activity. Figure 13-7 shows a sample access log configuration, which is automatically created during the basic configuration with the Wizard. See 2.3.1, “Your first HTTP Server (powered by Apache) via a wizard” on page 24.

Log	Attributes
Example logs/gif-requests.log	Log format: Common Environment variable condition: gif-image Expiration: 30 Days Maximum cumulative size: 10 Megabytes
Example logs/nongif-requests.log	Log format: Common Environment variable condition: lgif-image Expiration: 30 Days Maximum cumulative size: 10 Megabytes
logs/access_log	Log format: combined Environment variable condition: Expiration: 10 Days Maximum cumulative size: 10 Megabytes

Figure 13-7 Access log settings and log formats

What the setup means

Notice that our access log uses the *combined* format. It expires after 10 days and it is allowed to have a maximum cumulative size of 10 MB. That means, that, based on the General settings page, a new log file is created daily. It's allowed to grow up to 20 MB. If the server encounters a denial of service attack, it fills up the log file to its maximum size and stops logging. Otherwise it closes the log file and opens a new one. If the cumulative size of all access_log files is beyond the configured 10 MB, the server starts to delete the expired ones. If it's still too large, all other log files are deleted, beginning from the oldest, until the cumulative size is reached, or less.

Log files that are currently in use are not processed. The previous explanation is valid for all custom logs as well as for error and Fast Response Cache Accelerator (FRCA) logs.

Tip: There is no restriction on the maximum amount of customized logs you define, but consider that every log file (and log maintenance) produces overhead for your HTTP Server (powered by Apache).

Log management options

As shown in Figure 13-7, you can set an expiration period and a maximum cumulative size for log files. Following is an explanation of the two log management options:

- **Expiration:** This option specifies an integer value that indicates the number of days before a log file expires. Files older than this value are deleted. A value of 0 means the log file will never expire. The age of the log file is determined by the file creation date (as reported by the operating system). A log file that is currently open and active in the server instance is not deleted.

Note: By default, the HTTP server starts the deletion process every day at midnight. That means, if your server instance is not up and running at midnight, log maintenance will not occur. A new directive, which was not available via the IBM Web Administration for iSeries interface at the time of writing the redbook, can be used to change the time at which the server instance is running the log deletion process. The following example shows the directive that needs to be added to start the log maintenance process at 10:00 o'clock in the morning.

```
LogMaintHour 10
```

The deletion process starts at the top of the hour that is specified as a parameter to the LogMaintHour directive. Valid values are 0 - 23. Note that this directive is displayed in the GUI as not recognized.

- **Maximum cumulative size:** This option specifies an integer value indicating the maximum aggregate size of log files. When the combined log files exceeds this value in bytes, files are deleted starting with the oldest file. Log files are deleted until the cumulative size is within the specified value. A value of 0 means there is no size limit. If both the expiration and maximum cumulative size are configured to non-zero values, the expired log files are deleted first. If the maximum cumulative size is still exceeded after the expired files are deleted, the server continues deleting log files (oldest files first) until the cumulative size is achieved.

What is stored in these logs

Let's see what kind of information is stored in these three logs based on the log format we chose. Figure 13-8 shows five user-defined formats and the type of information that we want them to collect. Each of the case-sensitive tokens that we can use in a format definition represents different information about the client, the request received, or the status of client-server communications.

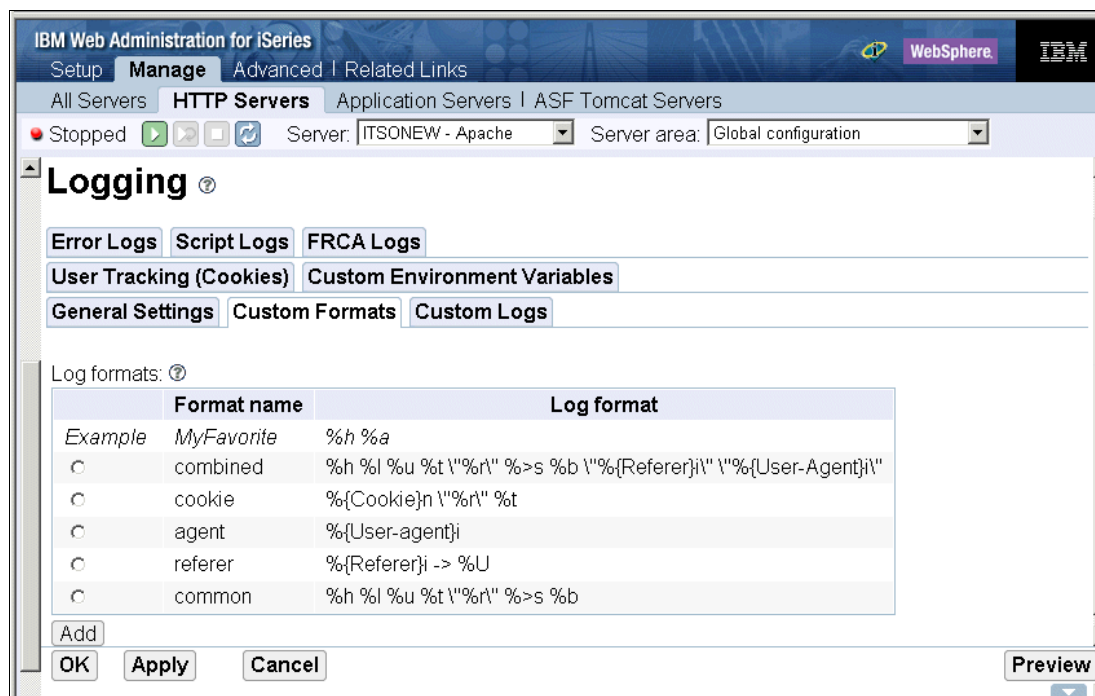


Figure 13-8 Log formats

We now analyze an access log entry, looking for the data that we included in the associated log format. Let's say we want to access our server's sample home page. We open a browser window and type `http://servername:port` in the address bar, as shown in Figure 13-9.

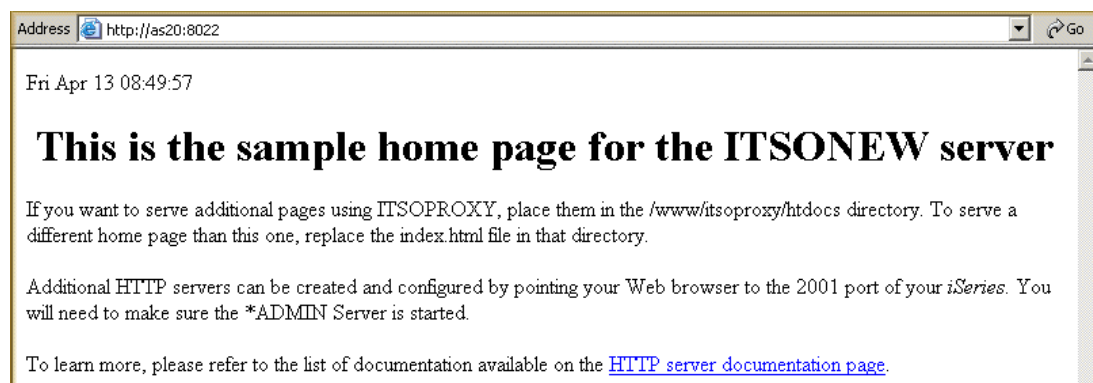


Figure 13-9 Sample home page

The last line in our server's access log now looks like the one in Figure 13-10.

```
Browse : /www/itsonew/logs/access_log
Record : 1 of 1 by 18          Column : 1 130 by 131
Control :

....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+....0....
9.146.221.156 - - 01/Oct/2004:13:11:11 +0200| "GET / HTTP/1.1" 200 717 "-" "Mozilla/4.0 (compatible; MS
*****End of Data*****
```

Figure 13-10 Analyzing the access log

Table 13-3 breaks down our string and identifies the relationship between what we chose to log in Figure 13-8 and the data our server collected.

Table 13-3 Understanding the access log

Access log	Token	Meaning
9.146.221.156	%h	The remote host's IP address.
-	%l	The user logged on to the remote system. In this case, the browser did not provide the user's name for security reasons.
-	%u	Name of the authenticated user (no user authentication was performed yet).
01/Oct/2004:13:11:11 +0200	%t	Date and time the request was served. +0200 is the difference from UTC. See system value QUTCOFFSET
"GET / HTTP/1.1"	%r	The request received. We can identify the method (GET), the Uniform Resource Identifier (URI) (/ , since we did not request any specific document) and the protocol used for this transaction (HTTP version 1.1).
200	%>s	The HTTP status code. 200 means that this transaction was successful. See 13.2.8, "HTTP status codes" on page 352, for more information.
717	%b	The amount of data transmitted, in bytes.
"-"	%{Referer}	The page we came from. There is none in this case, since we manually typed our address.
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT® 5.0)"	%{User-Agent}	Information about the browser and operating system on the remote client. In this case, Microsoft Internet Explorer on a Microsoft Windows 2000 operating system.

There is much more information you can log. For more information about customizing log formats, see the log format article in the HTTP Server documentation center at:

<http://www-1.ibm.com/servers/eserver/series/software/http/docs/doc.htm>

You should also see the Worldwide Web Consortium (W3C) logfile standards at:

<http://www.w3.org>

Error logs

Servers created using the GUI wizard always produce an error log by default. Look for error log files in the /logs subdirectory of your server root. Basic error logs are most useful for debugging configuration problems, such as when a document is not accessible or the URI (the path you add after the server address) you're pointing to is not working as expected. Error logs also keep track of configuration changes, server end/restart, and record some

system errors. Be aware that any problem detected after server initialization is most likely not recorded in the server job logs (unless a critical condition occurs), but in the error log. As you can see in Figure 13-11, error logs are really easy to configure. Figure 13-11 shows the configuration panel that results in the following directives in the configuration file:

- ▶ ErrorLog logs/error_log
- ▶ LogLevel warn

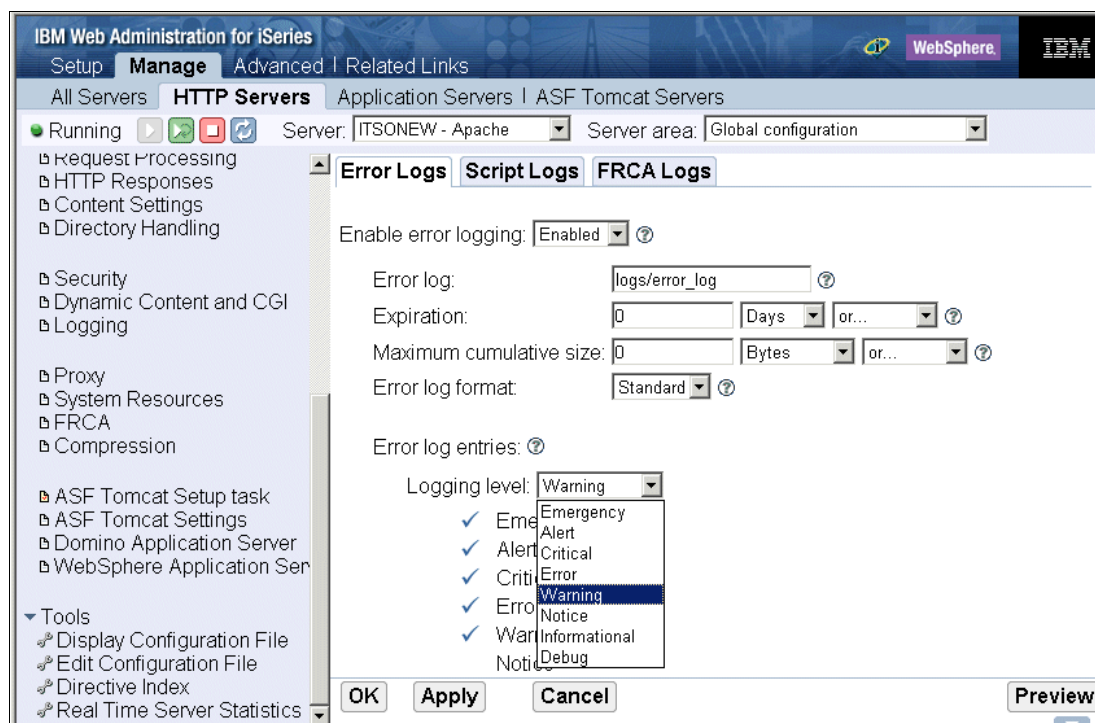


Figure 13-11 Error Logs settings and resulting directives

Note: The HTTP Server (powered by Apache) uses error logs per default. You may not be able to see the directives `ErrorLog` and `LogLevel` in the configuration file, although errors are being logged. To disable error logging, simply change the configuration directive to `ErrorLog Off` or use the GUI (Figure 13-11) and set it to `DISABLED`.

Script logs

Script logs record all CGI parsed data and, therefore, can have a significant impact on CGI performance. They should be used for debug purposes only and not be kept active all the time. Being a mere debug tool, they are not customizable, but saved for the maximum amount of data to be collected. The Script log settings in Figure 13-12 result in these directives:

- ▶ ScriptLog logs/script_log (see Script log)
- ▶ ScriptLogLength 10385760 (see Maximum log file size)
- ▶ ScriptLogBuffer 1024 (see Maximum log entry size)

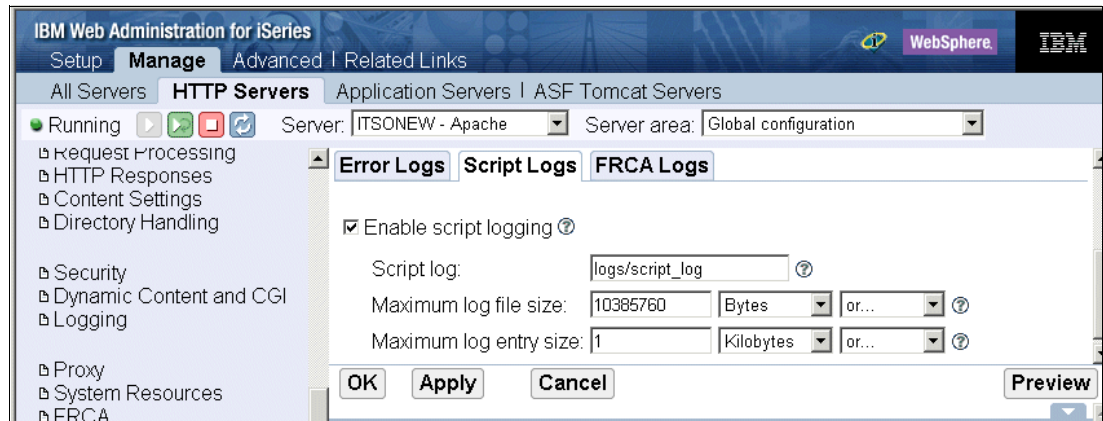


Figure 13-12 Script Logs settings

Custom formats

Each server configuration file contains information about the type of log files the server creates. Logging information allows you to track and generate reports on your server's activity. This page allows you to add and remove unique format names and their associated formats. After you define them, you can specify a format name on one or more CustomLog or FRCACustomLog directives. The format defines the information that is recorded with each entry in the log file. Figure 13-13 shows the benefit of using this page. The list provides an explanation for each token.

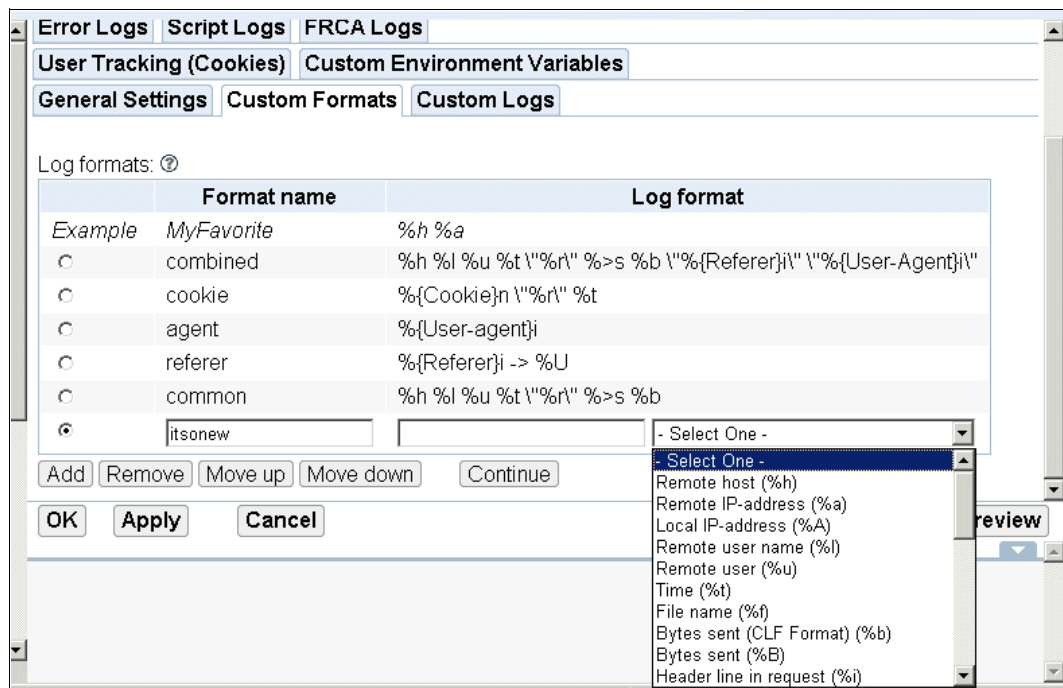


Figure 13-13 Custom Formats page

FRCA logs

This log is used to log FRCA requests to the server (Figure 13-14). You perform the setup in the same way as in “Custom logs” on page 332.

Tip: FRCA collects logging data in System Licensed Internal Code (SLIC), based on FRCAMaxCommTime and FRCAMaxCommBufferSize directives (see 10.6.6, “Miscellaneous FRCA directives beyond the online help” on page 296, for configuration details). When it sends the data to the HTTP Server (powered by Apache), which is above the Machine Interface (MI), this data comes as a “chunk”. The log files entries can be out-of-order and may be more difficult to read. All the log data is there, but not in the same order as the requests were processed.

This is done to improve the overall performance of the HTTP Server (powered by Apache) and FRCA servers.

For more information about FRCA, see 10.6, “Fast Response Cache Accelerator” on page 281.

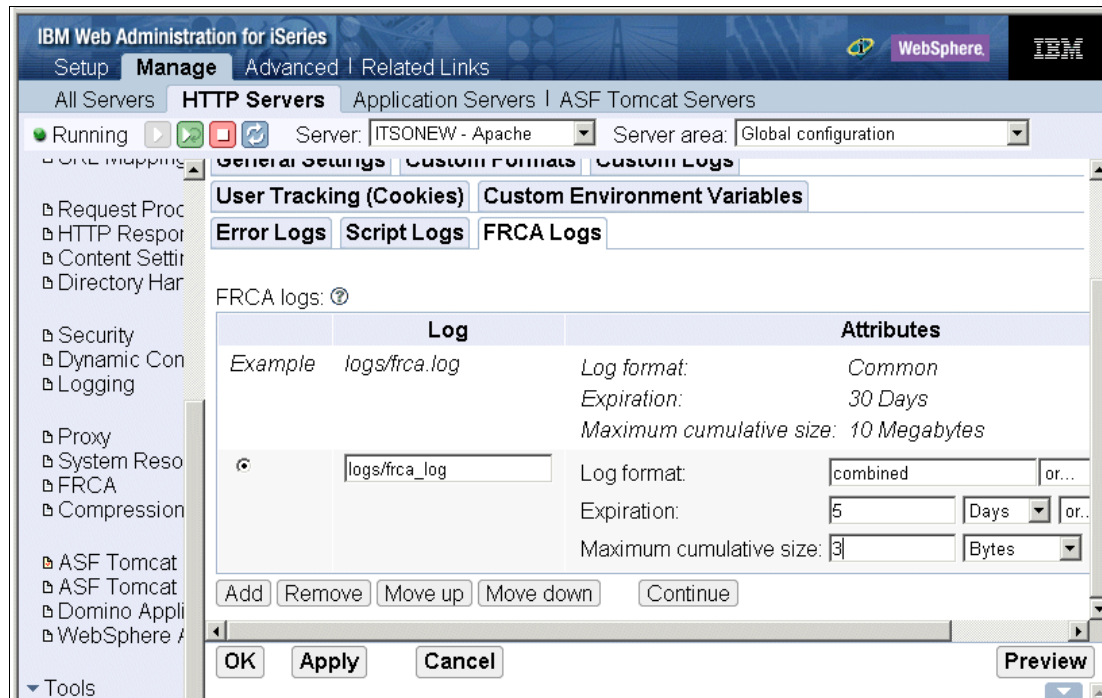


Figure 13-14 Setting up for FRCA logs

User tracking (cookies)

This page provides options for tracking user requests via client-side cookies (Figure 13-15).

The screenshot shows the IBM Web Administration for iSeries interface. The top navigation bar includes 'Setup', 'Manage', 'Advanced', and 'Related Links'. Below this, there are tabs for 'All Servers', 'HTTP Servers', 'Application Servers', and 'ASF Tomcat Servers'. The 'HTTP Servers' tab is selected, and the 'Server' dropdown is set to 'ITSONEW - Apache'. The 'Server area' dropdown is set to 'Global configuration'. The left sidebar contains a tree view with categories like 'Request Proc', 'HTTP Respor', 'Content Setti', 'Directory Har', 'Security', 'Dynamic Con', 'Logging', 'Proxy', 'System Reso', 'FRCA', 'Compression', 'ASF Tomcat', 'ASF Tomcat', 'Domino Appli', 'WebSphere A', 'Tools', and 'Display Confi'. The main content area is titled 'Logging' and has several sub-tabs: 'General Settings', 'Custom Formats', 'Custom Logs', 'Error Logs', 'Script Logs', 'FRCA Logs', 'User Tracking (Cookies)', and 'Custom Environment Variables'. The 'User Tracking (Cookies)' tab is selected. It contains the following settings: 'Track user requests in a cookie:' set to 'Enabled'; 'Cookie name:' set to 'Apache'; 'Expiration period:' set to '300' 'Seconds'; 'Cookie format:' set to 'Cookie'; and 'Domain to which the tracking cookie applies:' with an empty text box. A note states: 'Note: A custom log must be configured to collect any cookies your server is enabled to track'. At the bottom, there are buttons for 'OK', 'Apply', 'Cancel', and 'Preview'.

Figure 13-15 Setting up user tracking

Customer environment variables

This page allows you to add and remove environment variables and their associated attribute values (Figure 13-16).

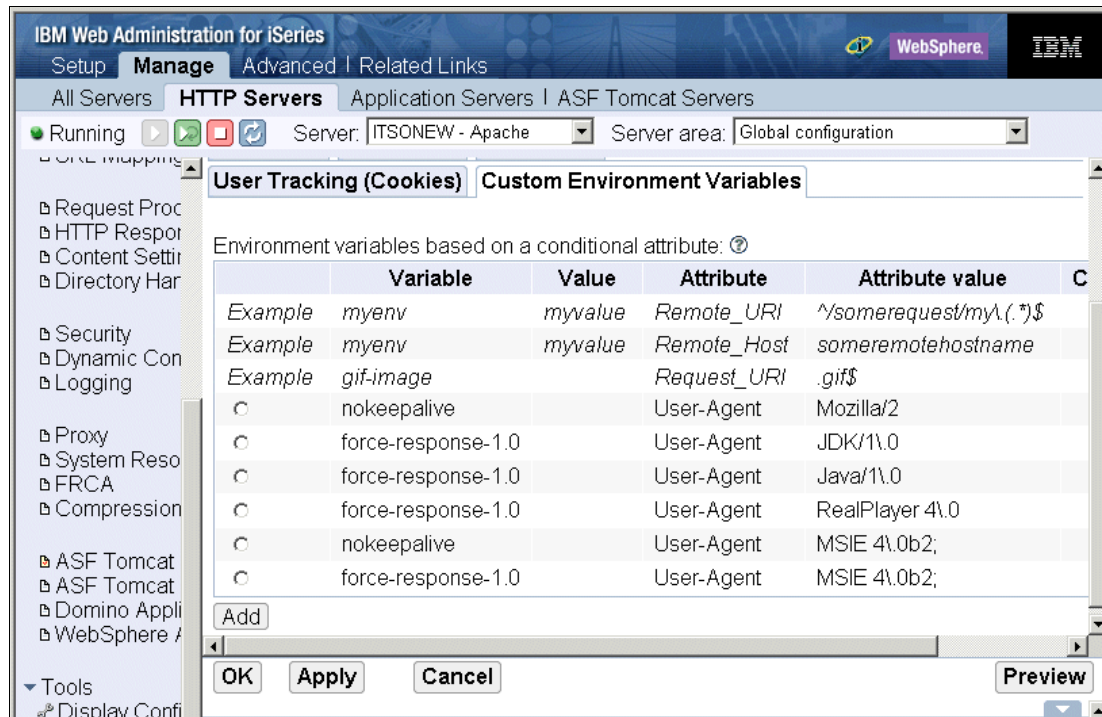


Figure 13-16 Adding customized environment variables

Log analyzers

You can extract access reports, usage statistic, and other interesting data from your HTTP Server (powered by Apache) logs. A wide range of both commercial and freeware products is available for this purpose, from the powerful IBM Tivoli Web Site Analyzer suite to simple log parsing scripts. See:

<http://www-306.ibm.com/software/tivoli/products/web-site-analyzer/>

13.2.4 Net.Data logs and traces

You can activate Net.Data traces and logs by adding the following directives to your Net.Data INI file:

```
DTW_ERROR_LOG_DIR [=] full_directory_path
DTW_ERROR_LOG_LEVEL [=] OFF | INFORMATION | ERROR | INFORMATION+ERROR | ALL
DTW_TRACE_LOG_DIR [=] full_directory_path
DTW_TRACE_LOG_LEVEL [=] OFF | APPLICATION | SERVICE
DTW_TRACE_MERGE_RECORDS [=] YES | NO
```

Remember that the server user profiles QTMHHTTP and QTMHHTTP1 need access to the Net.Data log folder.

Note: Net.Data logging and tracing support is available through the latest Net.Data or HTTP group PTFs. See the iSeries Net.Data Web site for updated information:

<http://www.iseries.ibm.com/netdata>

You should also refer to 7.3, “Net.Data: A ready-made scripting tool” on page 161.

13.2.5 HTTP server trace

An HTTP server trace provides additional information about server operations, from process management to URI interpretation. Server traces can be activated from the GUI Manage HTTP Servers display by starting the server with the lowercase -ve, -vi, and -vv startup parameters. The Start TCP/IP Server (STRTCPSVR) CL command also supports these startup options. You can collect the same data when the server is already active using the Trace TCP/IP Application (TRCTCPAPP) and Dump User Trace (DMPUSRTRC) commands. Be advised that this tracing facility does not support concurrent tracing of multiple HTTP servers.

Note: The HTTP Server (powered by Apache) does not support the -vi, -ve, and -vv switches on server restart. If you are unable to end all server jobs, use the Trace TCP/IP Application (TRCTCPAPP) command instead (see Figure 13-17).

Trace TCP/IP Application (TRCTCPAPP)		
Type choices, press Enter.		
TCP/IP application	> *HTTP	*FTP, *SMTPSVR, *SMTPCLT...
Trace option setting	*ON	*ON, *OFF, *END, *CHK
Maximum storage for trace . . .	*APP	1-16000, *APP
Trace full action	*WRAP	*WRAP, *STOPTRC
HTTP server instance	> MYSERVER	Character value
Trace level	*ERROR	*ERROR, *INFO, *VERBOSE
		Bottom
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display F24=More keys		

Figure 13-17 Trace TCP/IP Application (TRCTCPAPP): Turning on trace

The HTTP server trace can be set to operate at three different levels: *error*, *information*, and *verbose*. User trace data for both parent and child helper jobs is automatically dumped as soon as a failure condition is detected. The Dump User Trace (DMPUSRTRC) command is otherwise used to direct trace output to the same database file while server jobs are still active. Job name, number, and user profile for each one of your HTTP server jobs are required. Trace output is dumped to file QAP0ZDMP in QTEMP in a member called QP0Znnnnnn (where *nnnnnn* is the HTTP server job number you gave to the DMPUSRTRC command).

Table 13-4 shows the usage and purpose of this tracing facility.

Table 13-4 Service trace activation

Startup switch	TRCTCPAPP	Trace output
-ve	*ERROR	Server startup only. Nothing else is recorded unless an error occurs.
-vi	*INFO	Server startup and initialization, including directive processing, character conversion, and client request handling.
-vv	*VERBOSE	All the above plus application programming interface (API) and module invocation, HTTP headers, error messages.

Here are three examples of how an HTTP server trace can be collected in different environments:

- ▶ A test environment, an ideal situation in which you have complete control over the server
- ▶ A business-critical application, where the HTTP server is the core component of your On Demand Business infrastructure and must be available at any time
- ▶ Somewhere in between, a third scenario that fits in between the two extremes

A test environment

You are testing a stand-alone HTTP Server (powered by Apache) configuration that is not working as you expected. The server stopped and you are ready to collect an HTTP trace.

1. Start the HTTP server with the -ve, -vi, or -vv option. Look for completion message CPCA984 (see Figure 13-18) in the server job log for confirmation that the trace option you specified was accepted.

Additional Message Information			
Message ID	: CPCA984	Severity	: 00
Message type	: Completion		
Date sent	: 10/09/01	Time sent	: 09:09:09
Message : User Trace option changed for job 032335/QTMHHTTP/ITSOSRV1.			
			Bottom
Press Enter to continue.			
F3=Exit F6=Print F9=Display message details F12=Cancel F21=Select assistance level			

Figure 13-18 Additional Message Information: CPCA984 - A successful activation

2. Reproduce the failure. Skip the next step if the server jobs already ended.
3. Stop the HTTP server.

4. Use the Work with Spooled Files (WRKSPLF) CL command or iSeries Navigator (Operations Navigator for V5R1) to retrieve QZSRHTTPTR spooled files for the QTMHHTTP user profile (see Figure 13-19).

Work with All Spooled Files								
Type options, press Enter.								
1=Send 2=Change 3=Hold 4=Delete 5=Display 6=Release 7=Messages								
8=Attributes 9=Work with printing status								
Opt	File	User	Device or Queue	User Data	Sts	Total Pages	Cur Page	Copy
	QZSRHTTPTR	QTMHHTTP	PRT01	QSRV035027	HLD	28		1
	QZSRHTTPTR	QTMHHTTP	PRT01	QSRV035029	HLD	29		1
	QZSRHTTPTR	QTMHHTTP	PRT01	QSRV035028	HLD	44		1
								Bottom
Parameters for options 1, 2, 3 or command								
====> WRKSPLF SELECT(QTMHHTTP)								
F17=Top F18=Bottom F21=Select assistance level F24=More keys								

Figure 13-19 Work with All Spooled Files: Startup switch output

A business-critical application

The HTTP Server (powered by Apache) is a business-critical application running full time. It cannot be stopped. You are experiencing occasional problems and need a server trace to identify the source of your troubles.

1. Start the trace using the command:

```
TRCTCPAPP APP(*HTTP) SET(*ON) HTTPSVR(SERVERNAME) TRCLVL(*VERBOSE)
```

Select an appropriate level of information. Message CPC1129 (see Figure 13-21) is posted to the server job log.

2. Reproduce the failure. Skip the next step if server jobs already ended.
3. Stop the trace with the command:

```
TRCTCPAPP APP(*HTTP) SET(*OFF)
```

4. Use the Work with Spooled Files (WRKSPLF) CL command or iSeries Navigator to retrieve the QZSRHTTPTR spooled files for the user profile you used for signon (see Figure 13-20).

Work with All Spooled Files									
Type options, press Enter.									
1=Send 2=Change 3=Hold 4=Delete 5=Display 6=Release 7=Messages									
8=Attributes 9=Work with printing status									
Opt	File	User	Device or Queue	User Data	Sts	Total Pages	Cur Page	Copy	
	QZSRHTTPTR	GBANCHELLI	PRT01	QSRV035077	HLD	1		1	
	QZSRHTTPTR	GBANCHELLI	PRT01	QSRV035078	HLD	1		1	
	QZSRHTTPTR	GBANCHELLI	PRT01	QSRV035076	HLD	1		1	
									Bottom
Parameters for options 1, 2, 3 or command									
===>									
F3=Exit F10=View 4 F11=View 2 F12=Cancel F22=Printers F24=More keys									

Figure 13-20 TRCTCPAPP output

Somewhere in between

You are developing an intranet application, but you constantly run into an error. The server cannot be stopped, but you are free to choose your debugging options.

1. Start the trace using the command:

```
TRCTCPAPP APP(*HTTP) SET(*ON) HTTPSVR(SERVERNAME) TRCLVL(*VERBOSE)
```

Select an appropriate level of information. Message CPC1129 (see Figure 13-21) is posted to the server job logs.

Additional Message Information			
Message ID	CPC1129	Severity	00
Message type	Completion		
Date sent	10/11/01	Time sent	17:30:11
Message : Job 035076/QTMHHTTP/GERONIMO changed by GBANCHELLI.			
			Bottom
Press Enter to continue.			
F3=Exit F6=Print F9=Display message details F12=Cancel			
F21=Select assistance level			

Figure 13-21 CPC1129: The trace is active

2. Reproduce the error.

3. At this point, you want to examine the data collected so far, but you still need to keep tracing server activity. Retrieve job numbers for your HTTP server jobs and feed them to the DMPUSRTRC command. Message CPCA986 is posted to your job log (see Figure 13-22).

```
Additional Message Information

Message ID . . . . . : CPCA986      Severity . . . . . : 00
Message type . . . . . : Completion
Date sent . . . . . : 10/11/01      Time sent . . . . . : 17:31:43

Message . . . . . : User Trace data for job 035076/QTMHHTTP/GERONIMO dumped to
                    member QP0Z035076 in file QAP0ZDMP in library QTEMP.
Cause . . . . . : The User Trace records associated with job
                  035076/QTMHHTTP/GERONIMO were successfully dumped to member QP0Z035076 in
                  file QAP0ZDMP in library QTEMP.

                                                    Bottom

Press Enter to continue.

F3=Exit  F6=Print  F9=Display message details  F12=Cancel
F21=Select assistance level
```

Figure 13-22 CPCA986: Trace data has been dumped

4. Look for the file QAP0ZDMP in QTEMP. This file contains a QP0Znnnnnn member for each one of the server job numbers (nnnnnn) you used in the DMPUSRTRC command.

Tips: Never forget to stop the trace with TRCTCPAPP APP(*HTTP) SET(*OFF) when it is no longer needed. Also remember that you can access only the content of the QTEMP library from your current session. It is discarded as soon as you sign off.

13.2.6 Collection Services performance data

Web-based transaction processing and Web-serving environments continue to grow in importance and complexity. Your HTTP Server (powered by Apache) is the focal point of many different kinds of On Demand Business environments including SSL, cache accelerators such as FRCA, and WebSphere Application Server.

Tip: The iSeries HTTP Server (powered by Apache) does not support mod_status. This simple module allows a Web administrator to take a picture of an Apache server and see performance-related statistics that drill down to the work performed by each individual thread.

mod_status was adjusted to work with the Apache 2.0 threaded server by the Apache Software Foundation (ASF). However the fact that iSeries implements asynchronous input/output (I/O) (see 10.2.1, “Threads and asynchronous I/O” on page 228) provides complex challenges for implementing mod_status on the iSeries server.

Unique to the iSeries, HTTP server statistics are saved into collection services in V5R2. The advantage on the iSeries server is that these reports can provide a more holistic view of system performance. For example, it helps in situations where you may say, “Ah, I see the reason that the HTTP Server (powered by Apache) is running so slow. That programmer recompiled the entire LOB application again on the production server!”

Beginning in V5R2, you can define your own performance collection categories with iSeries Collection Services. The HTTP Server (powered by Apache) uses this new feature to integrate performance data into Collection Services.

As shown Figure 13-23, the Standard plus protocol profile (the default used by Collection Services) automatically collects HTTP Server (powered by Apache) if Collection Services detects this application server is active on the system. As with all Collection Services “collection object data”, the new statistics are placed into the following files via the currently available iSeries Navigator “Create performance database files” function or the OS/400 Create Performance Data (CRTPFRTA) command:

- ▶ QAPMHTTPPB: Contains the basic data for HTTP Server (powered by Apache)
- ▶ QAPMHTTPPD: Contains detailed data for HTTP Server (powered by Apache)

HTTP data collection category to contain HTTP performance data for Collection Services. The HTTP performance data can then be queried to analyze HTTP server activity and better understand what types of HTTP transactions are being processed by the iSeries (for example, static files, CGI, or Java Servlets).

In addition, V5R2 Performance Tools for iSeries, 5722-PT1, has new sections in the System and Component Reports for HTTP statistics.

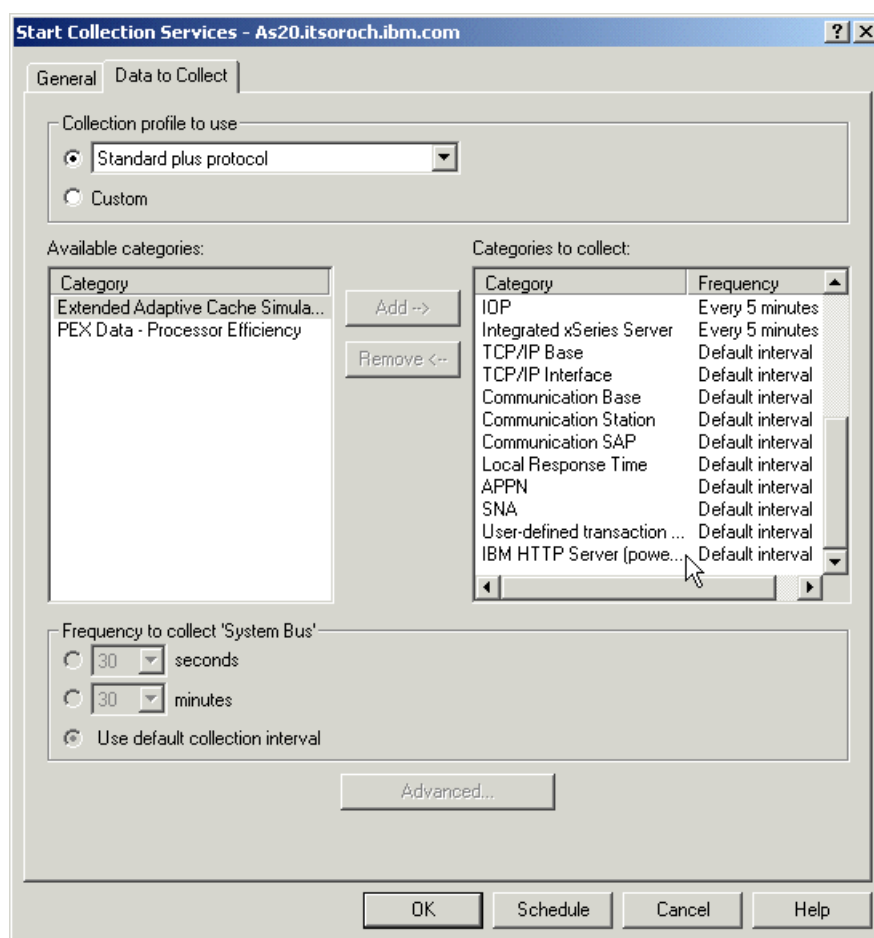


Figure 13-23 Start Collection Services: Data to Collect page

The types of data collected are broken down into the following ways. Note that this is per server job and the statistics are shown for each interval and request type within the interval.

- ▶ **SR:** Requests handled internally by the server itself. No program processing is necessary.
- ▶ **SL:** Requests of all types received via SSL. Reports activity that occurred over an SSL connection even though that activity is also reported with other applicable request types.
- ▶ **PX:** Proxy requests.
- ▶ **CG:** CGI requests.
- ▶ **WS:** WebSphere requests.
- ▶ **JV:** IBM Java Servlet Engine requests.
- ▶ **UM:** Requests handled by user modules.
- ▶ **FS:** Static requests handled by FRCA.
- ▶ **FX:** Requests proxied by FRCA.

Starting Collection Services for the HTTP Server (powered by Apache)

We use iSeries Navigator to start and work with Collection Services. To start Collection Services, as shown in Figure 13-23, follow these steps:

1. Log on to your iSeries server using with iSeries Navigator.
2. Expand ***your server*** → **Configuration and Service**.
3. Right-click **Collection Services** and select **Start Collecting...**
4. You see the Start Collection Services window (Figure 13-24). For the most part, you may accept all the defaults.

Take note of the library to store collections since you need to know this later to find the files. Also, we chose to set the Default collection interval for detailed data to 5 minutes.

5. Click **OK** to start Collection Services.

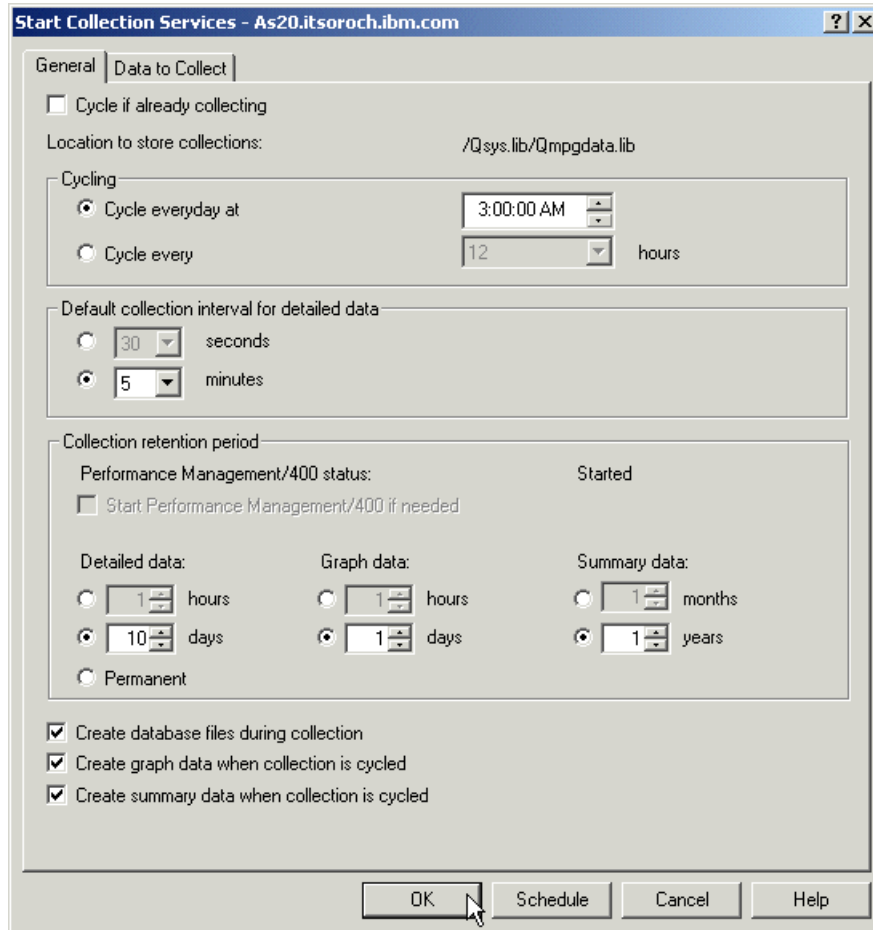


Figure 13-24 Start Collection Services: General page

At this point, your HTTP Server (powered by Apache) server and related applications should be up and running in a “steady state”. Collect the data for as long as you need.

When you are done collecting the data, stop Collection Services:

1. Right-click **Collection Services** and select **Stop Collecting...**
2. In the Stop Collection Services panel, click **OK**.

Performance Tools reports

Performance Tools for iSeries were enhanced in V5R2 to generate reports based on the HTTP performance data. The reports contain information about the transactions processed by HTTP server jobs. Follow these steps to see the System and Component reports for the HTTP server:

1. From a 5250 session, enter the Start Performance Tools (STRPFRT) command.
2. Select option 3 (Print performance report).
3. Change the Library to QMPGDATA. Or, specify the library in which you saved the collection data. Press Enter to refresh the list of collections.

4. In the Print Performance Report - Sample data display (Figure 13-25), type option 1 (System report) to the left of the collection data member. Use the Date and Time columns to make sure you select the correct one.

Print Performance Report - Sample data				
Library QMPGDATA				
Type option, press Enter.				
1=System report 2=Component report 3=Job report 4=Pool report				
5=Resource report				
Option	Member	Text	Date	Time
1	Q168151618		06/17/03	15:16:18
	Q168125025		06/17/03	12:50:25
	Q143095558		05/23/03	09:55:58

Figure 13-25 Performance Tools: Option 1 (System report)

5. In the Select Sections for Report display, press F6 to print the entire report.
6. In the Select Categories for Report display, press F6 to print the entire report.
7. In the Specify Report Options display, enter a meaningful report title.

Tip: We like to copy this System Report title so we can paste it to the Component Report title later. This allows us to match these pairs of reports in the future.

8. Press Enter to submit this work to the batch queue.
9. In the Print Performance Report - Sample data display (Figure 13-25), type option 2 (Component report).
10. Repeat steps 5 through 8 for the Component report.

When the batch jobs finish, you should have two new spool files in OUTQ QPFROUTQ. Figure 13-26 shows the output of the System Report.

Display Spooled File											
File QPPTSYSR								Page/Line		9/1	
Control +5								Columns		1 - 130	
Find											
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...0...+...1...+...2...+...3											
System Report								052303		10:22:0	
HTTP Server Summary								Page 000			
Batch jobs											
Member . . . : Q143095558 Model/Serial . . : 270/10-4RT9M				Main storage . . : 8000.0 MB				Started :		05/23/03 09:55:5	
Library . . . : QMPGDATA System name . . : ASM27				Version/Release : 5/ 2.0				Stopped :		05/23/03 10:15:0	
Partition ID : 000 Feature Code . . : 22AB-2253-1520											
Server name	Server job user	Server job number	Server start date/time	----- Threads -----	Active	Idle	-- Inbound Connections --	Non-SSL	SSL	Requests received	Responses sent
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
ADMIN	QTMHHTTP	050851	05/22/03 11:04	0	40		0	0		0	0
HATSLEHTTP	QTMHHTTP	051068	05/22/03 11:16	0	40		0	0		0	0
IAXHTTPEXA	QTMHHTTP	051477	05/23/03 08:47	0	40		0	0		0	0
IAXHTTPEXB	QTMHHTTP	051045	05/22/03 11:15	1	39		39	0		39	39
IAXHTTSSL	QTMHHTTP	051352	05/23/03 08:45	0	40		0	321		321	321
IWA	QTMHHTTP	051022	05/22/03 11:13	0	40		0	0		0	0

Figure 13-26 Performance Tools: System Report - HTTP Server Summary

Figure 13-27 shows the output of the Component Report for the HTTP server activity. This example is for the same HTTP server - IAXHTTPSSL included in the System Report HTTP statistics example. Here you can see that, of the 321 requests shown to be processed by this server in the System Report example, 308 were processed during the 5 minute interval ended at 10:00. Thirteen were processed during the 5 minute interval ended at 10:05. You can also see the average “K bytes” (1024 bytes) sent each second was 5 KB and received was 1 KB during the 10:00 interval.

Looking closely, you can tell via the SL Request Type (all requests handled under an SSL connection) that all such requests were handled by some application running under a WebSphere Application Server (WS Request Type). As defined on the next page, SL counts also appear under another request type. You have to adjust to this accounting method to know that actually 321 requests were received from browsers, not the 642 (2 x 321) shown as total requests received and responses sent.

This example shows that no response was rejected or considered “in error”. High error or reject values need to be investigated.

Note also the algorithm used when computing average K bytes per second. The Performance Tools code knows that SL and WS values represent duplicates of each other. Using our 2 intervals example, we have 5K Bytes and 0K bytes per second transmitted averaged as 2K bytes per second and 1K Bytes and 0K bytes per second received averaged as 0 K bytes per second.

Component Report								5/23/03 10:22:0	
HTTP Server Activity								Page 2	
Batch jobs									
Member : Q143095558 Model/Serial . : 270/10-4RT9M				Main storage . . : 8000.0 MB		Started : 05/23/03 09:55:5		5/23/03 10:15:0	
Library . . : QMPGDATA System name . . :ASM27				Version/Release : 5/ 2.0		Stopped : 05/23/03 10:15:0			
Partition ID : 000 Feature Code . :22AB-2253-1520									
Server : 051352/QTMHHTTP/IAXHTTPSSL									
----- Requests ----- Responses -----									
Itv	Req	Pct			Pct			KB	KB
End	type	Received	Rejected	Rejected	Sent	Error	Error	Transmitted /Second	Received /Second
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
10:00	SL	308	0	.00	308	0	.00	5	1
10:00	WS	308	0	.00	308	0	.00	5	1
10:05	SL	13	0	.00	13	0	.00	0	0
10:05	WS	13	0	.00	13	0	.00	0	0
Column		Total			Average				

Requests Received					642				
Requests Rejected					0				
Pct Requests Rejected					.000				
Responses Sent					642				
Responses in error					0				
Pct Responses in error					.000				
KB Transmitted/Second					2				
KB Received/Second					0				
-- Interval end time (hour and minute)									
-- The type of request being reported									
-- The number of requests received by the server									
-- The number of requests received that were not valid									
-- Percentage of requests received that were not valid									
-- The number of responses sent									
-- The number of responses in error									
-- Percentage of responses in error									
-- Number of kilobytes (1024 bytes) transmitted per second									
-- Number of kilobytes (1024 bytes) received per second									

Figure 13-27 Performance Tools: Component Report - HTTP Server Activity

For more information

For further study of this topic, refer to the following resources:

- ▶ The iSeries Information Center has a starting point for performance-related topics that include the logging of information with iSeries Collection Services:
<http://publib.boulder.ibm.com/series/v5r2/ic2924/info/rzahx/rzahxebushttp.htm>
- ▶ Two Redbooks deal with collection services at V5R1. They do not have specific information about the HTTP Server (powered by Apache) information that is collected at V5R2.
 - *Managing OS/400 with Operations Navigator V5R1 Volume 1: Overview and More*, SG24-6226
 - *Managing OS/400 with Operations Navigator V5R1 Volume 5: Performance Management*, SG24-6565

13.2.7 Other startup parameters

Server startup parameters can also aid in problem determination and in testing your server configuration. In addition to the more debug-oriented `-ve`, `-vi`, `-vv` startup parameters discussed in 13.2.5, “HTTP server trace” on page 341, the following options are available:

Tip: These parameters are case sensitive.

- ▶ **-netccsid [nnn]**: Overrides the DefaultNetCCSID directive.
- ▶ **-fscsid [nnn]**: Overrides the default DefaultFsCCSID directive.
- ▶ **-d [serverroot]**: Sets the initial value for the ServerRoot variable to serverroot. This can be overridden by the ServerRoot directive.
- ▶ **-f [configuration]**: Uses the values in the *configuration* variable on startup. If the configuration does not begin with a `/`, then it is treated as a path relative to the ServerRoot. For example, the following command advises your server to use the configuration file from the fully qualified path that is specified:

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(ITSONEW '-f /www/apachedft/conf/httpd.conf')
```

The following command loads the file from the [serverroot]/conf directory:

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(ITSONEW '-f conf/httpd.conf')
```

- ▶ **-C [directive]**: Processes the given directive as though it was part of the configuration.
- ▶ **-c [directive]**: Processes the given directive after reading all the regular configuration files.
- ▶ **-V**: Displays the base version of the server, build date, and a list of compile time settings to your 5250 session. You may use the Page Up and Page Down keys to review the information. Press Enter to quit the terminal session. The server is *not* started. Here is an example printout:

```
Server version: Apache/2.0.43
Server built:   Nov 26 2002 15:57:01
Server's Module Magic Number: 20020903:0
Architecture:  128-bit
Server compiled with....
-D APR_HAS_SENDFILE
-D NO_LINGCLOSE
-D APR_USE_FCNTL_SERIALIZE
-D APR_USE_PTHREAD_SERIALIZE
-D APR_PROCESS_LOCK_IS_GLOBAL
```

```

-D APR_HAS_OTHER_CHILD
-D APR_CHARSET_EBCDIC
-D APACHE_XLATE
-D HTTPD_ROOT="/QIBM/UserData/HTTPD"
-D AS400
-D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
-D DEFAULT_ERRORLOG="logs/error_log"
-D AP_TYPES_CONFIG_FILE="conf/mime.types"
-D SERVER_CONFIG_FILE="conf/httpd.conf"
Press ENTER to end terminal session.

```

- **-l:** Displays a list of all modules compiled.
- **-t:** Tests the configuration file syntax but does not start the server. This command checks to see if all DocumentRoot entries exist and are directories. The command `STRTCPSVR SERVER(*HTTP) HTTPSVR(ITSONEW '-t')` results in:

```

Syntax OK
Press ENTER to end terminal session.

```

13.2.8 HTTP status codes

In some cases, your HTTP server responds with standard three-digit codes, such as 404, 500, and so on to a client request. These are known as *HTTP status codes*. They often provide additional information about the cause of a failure. Table 13-5 offers a quick guide on HTTP status codes.

Table 13-5 *HTTP status codes*

Status code group	Meaning	Example
1xx	Informational. Contains a provisional response that influences the client's behavior.	101 Switching Protocols Sent when an updated version of the HTTP protocol has been negotiated.
2xx	Successful. Also returns information about the status of negotiations and transactions.	200 OK The client request is fulfilled. Can contain additional information in the reply to GET, POST, HEAD, TRACE methods.
3xx	Redirection. Requests an action from the user agent.	304 Not Modified Often used in an answer to conditional requests, usually when caching is involved.
4xx	Client error. They are sent to your browser window when the page you requested cannot be served.	404 Not Found The server is unable to find anything matching the client request. Also used for security reasons when the requesting user should not have access to a particular resource.
5xx	Server error. Sent to your browser window when the server is unable to fulfill your request because of an internal problem.	503 Service Unavailable The server has encountered a temporary condition that is preventing the fulfillment of a client request.

For more information about HTTP status codes and their meaning, refer to Request for Comments (RFC) 2616 on the HTTP 1.1 protocol standard.

13.2.9 Communications trace

A communications trace is also a powerful tool for problem determination. It is useful for gathering information about the connection status and response time, and if you are experiencing troubles with the encoding of your files.

Here is a quick example of using a communications trace to all the IP datagrams received and sent from the iSeries point of view. The 5250 communication trace commands used are:

- ▶ Start Communications Trace (STRCMNTRC)
- ▶ End Communications Trace (ENDCMNTRC)
- ▶ Dump Communications Trace (DMPCMNTRC)
- ▶ Print Communications Trace (PRTCMNTRC)
- ▶ Delete Communications Trace (DLTCMNTRC)

This assumes that your configuration line object's name is ETHLINE. Enter the commands and perform the following actions in the order shown:

1. Enter the following command:

```
STRCMNTRC CFGOBJ(ETHLINE) CFGTYPE(*LIN) MAXSTG(256K) USRDTA(*MAX)
```

2. Start your HTTP Server (powered by Apache) and Web client and test your Web application. Keep your work at a minimum to lessen the amount of data collected.

3. Enter the following command:

```
ENDCMNTRC CFGOBJ(ETHLINE) CFGTYPE(*LIN)
```

4. The next step is to output the trace information. You have two options:

- Print the trace in to a spooled file using the following command:

```
PRTCMNTRC CFGOBJ(ETHLINE) CFGTYPE(*LIN) CODE(*ASCII) FMTBCD(*NO)
```

This option prints the trace in MAC layer format. If you want to print, for example, a trace in IP formatting for server port 80, you could enter the following command:

```
PRTCMNTRC CFGOBJ(ETHLINE) CFGTYPE(*LIN) CODE(*ASCII) FMTCPP(*YES) SLTPORT(80)  
FMTBCD(*NO)
```

Note: The spooled file of the communications trace output contains on the right side a so called eye-catcher. This is an area where all text in a trace is formatted in a readable format. However, when using the PRTCMNTRC command with the options as previously shown, all text, whether uppercase or lowercase, will be displayed in the spooled file in uppercase. In certain situations, it is necessary to consider the case. The following command flow creates a spooled file where case-sensitivity is considered.

- The following commands create a spooled file where uppercase and lowercase characters are displayed correctly in the eye-catcher area.

```
DMPCMNTRC CFGOBJ(ETHLINE) CFGTYPE(*LIN) TOSTMF('/barlen/trace010ct04')  
PRTCMNTRC FROMSTMF('/barlen/trace010ct04') CODE(*ASCII) SLTPORT(80)
```

This example also formats the trace for IP and contains only data for port 80.

5. Enter the Work with Job (WRKJOB) command and select option 4 (Work with spooled files). This is a fast way to find the spool file created by the PRTCMNTRC command.

6. Enter the following command:

```
DLTCMNTRC CFGOBJ(ETHLINE) CFGTYPE(*LIN)
```

For more information about how to take a communications trace, refer to IBM iSeries Support Line Knowledge Base document *TCP/IP Communications Trace Instructions*, 23825849.

13.2.10 Additional resources

For more information, consult the following sources:

- ▶ RFC Index

<http://www.rfc-editor.org/rfc.html>

- ▶ iSeries Support Line Knowledge Base

http://www-912.ibm.com/s_dir/slkbase.nsf/slkbase



High availability

If Web serving is a critical aspect of your business, you may want a highly available Web server environment. Highly available HTTP servers take advantage of iSeries clustering technology and make it possible to build a highly available Web site. This improves the availability of business-critical Web applications built with static Hypertext Markup Language (HTML) pages or Common Gateway Interface (CGI) programs. This enhancement is available with both the HTTP Server (powered by Apache) in addition to the HTTP Server (original).

Highly available HTTP servers are not supported on versions prior to V5R1.

14.1 Highly available Web server cluster on the HTTP server

The Web server cluster solution can provide:

- ▶ *Planned downtime*: If a Web server requires planned maintenance, it is possible to transfer the work to another node without visible service interruptions to the client.
- ▶ *No unplanned downtime*: If a machine fails, the work is transferred to another node with no human involvement and without visible service interruptions to the client.
- ▶ *Scalability*: When employing multiple nodes, it is possible to distribute the Web site workload over the cluster nodes.

Clusters are a collection of complete systems that work together to provide a single, unified computing capability.

A *liveness monitor* checks the state of the Web server. It interacts with the Web server and the clustering resource services in the event that a Web server fails (failover), or a manual switchover takes place (ensures no interruption of Web server services).

You can use the *clustered hash table* (part of the state replication mechanism) to replicate highly available CGI program state data across the cluster nodes. That way the state data is available to all nodes in the event that a Web server fails (failover) or is switched-over manually (switchover). To take advantage of this capability, an existing CGI program must be enabled in a highly available Web server environment. CGI programs write to the CGI application programming interfaces (APIs) to indicate what data is replicated. See *HTTP Server for iSeries Programming*, GC41-5435.

Three Web server cluster models are supported:

- ▶ Primary or backup with takeover Internet Protocol (IP) model
- ▶ Primary or backup with a network dispatcher model
- ▶ Peer model

14.1.1 Primary or backup with takeover IP model

In this model, the Web server runs on the primary and all backup nodes. The backup node or nodes are in an idle state, ready to become the primary Web server should the primary Web server fail (failover) or a switchover takes place. All client requests are always served by the primary node. Figure 14-1 illustrates a primary or backup with takeover IP model.

When the primary node fails (failover), or is brought down by the administrator, the failover/switchover process begins. The following steps are performed during failover/switchover:

1. One of the backup servers becomes the primary (the first backup in the switchover order).
2. Client requests are redirected to the new primary node. Assuming this client was not in the process of running a persistent CGI application, the failover is completely transparent.

Tip: You can provide a high availability (HA) environment with two iSeries servers, each with one instance of the HTTP Server (powered by Apache). Each of these instances serves the identical (a copy) HTML and images and from identical (a copy) httpd.conf configuration files. You can do this with a simple configuration. You do not need third-party software. This is assuming, however, that you are not providing a HA environment for your CGI application.

3. If the new primary receives a user request that belongs to a long-running-session (a CGI program that is updated to be a highly available CGI program), the server restores the request's state. The new primary retrieves that highly available CGI program's state information from the clustered hash table. The clustered hash table is part of the state replication mechanism.

Most non-HA CGI applications behave in the following manner:

- a. The client clicks the Submit button to send a new request to the Web server and your CGI application.
- b. Your persistent CGI application “wakes up”. Its state information is saved in static variables to determine what happened in the past with this client and what to do now. Parameters found in the URL are parsed and actions are taken. New state information is saved in the static variables for the next time this client returns to this iSeries server. HTML code is generated and sent to standard out for presentation to the remote client.
- c. The above “cycle” continues until the entire transaction is complete.

Most HA CGI applications behave in the following manner:

- a. The client clicks the Submit button, which sends a new request to the Web server and your CGI application.
 - b. Your persistent CGI application “wakes up”. Its state information is saved in the clustered hash table. It reads from the clustered hash table and updates local variables to determine what happened in the past with this client and what it must do now. Parameters found in the URL are parsed and actions are taken. New state information is written to the clustered hash table for the next time this same client returns to this (or the backup) iSeries server. HA support ensures that the information written to the clustered hash table on one iSeries server is replicated to the backup server. HTML code is generated and sent to standard out for presentation to the remote client.
 - c. The above “cycle” continues until the entire transaction is complete.
4. After the failed node recovers, you can restart the highly available Web server instance, which then becomes the backup system. If the system administrator wants the failed node to become primary again, they must perform a manual switchover. They can accomplish this with the IBM Simple Cluster Management interface available through iSeries Navigator (Operations Navigator in V5R1), a 5250 CL interface, or a business partner tool.

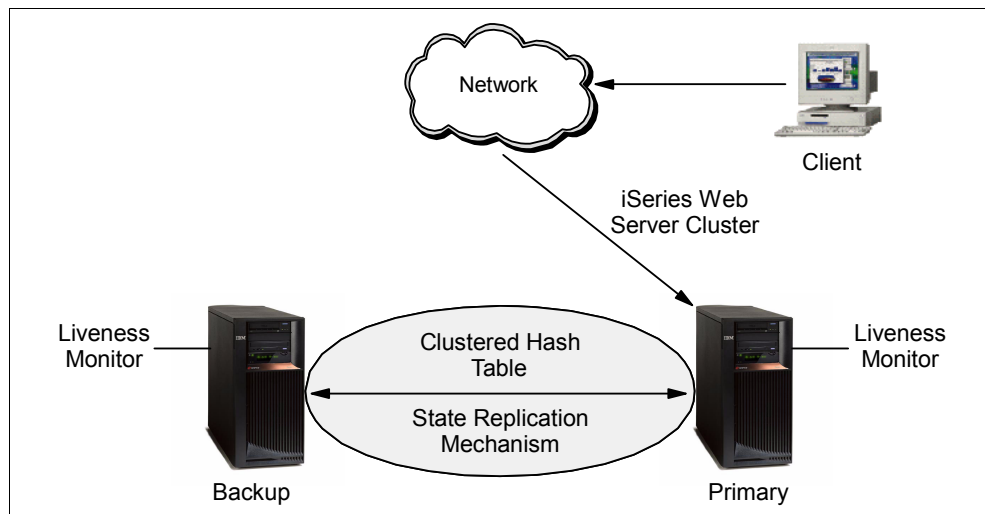


Figure 14-1 High availability: Primary or backup with takeover IP model

For an example, see 14.2, “A working primary or backup with takeover IP model” on page 359.

14.1.2 Primary or backup with a network dispatcher model

In this model, as with the primary or backup with takeover IP model, the Web server runs on the primary and all backup nodes. The backup nodes are in an idle state and all client requests are served by the primary node. A network dispatcher, such as the IBM WebSphere Edge Server, sends client requests to the Web server.

When the primary node fails (failover), or a switchover takes place, the failover/switchover process begins. The following steps are performed during failover/switchover:

1. One of the backup servers becomes the primary (the first backup in the switchover order).
2. The client requests are sent to the new primary node by the network dispatcher.
3. If the new primary receives a user request that belongs to a long-running-session, the server needs to restore the request's state. The new primary searches for the state either locally or in the clustered hash table. The clustered hash table is part of the state replication mechanism.
4. After the failed node recovers, the system administrator can restart the Web server instance and it becomes a backup Web server. If the system administrator wants the failed node to become primary again, they must perform a manual switchover.

Note: A node can join a recovery domain as a primary only if the Cluster Resource Group (CRG) is in inactive mode.

Figure 14-2 illustrates a primary or backup with a network dispatcher model.

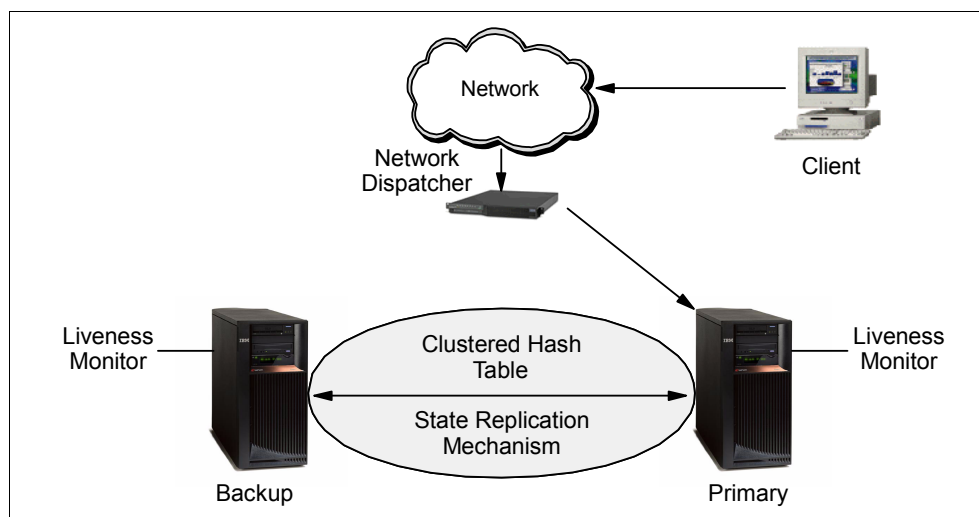


Figure 14-2 High availability: Primary or backup with a network dispatcher model

14.1.3 Peer model

In this model, there is no declared primary node. All nodes are in an active state and serve client requests. A network dispatcher, such as IBM WebSphere Edge Server, evenly distributes requests to different cluster nodes. This guarantees distribution of resources in case of a heavy load. Linear scalability is not guaranteed beyond a small number of nodes. After some nodes are added, scalability can disappear, and the cluster performance can deteriorate.

Figure 14-3 illustrates the peer model.

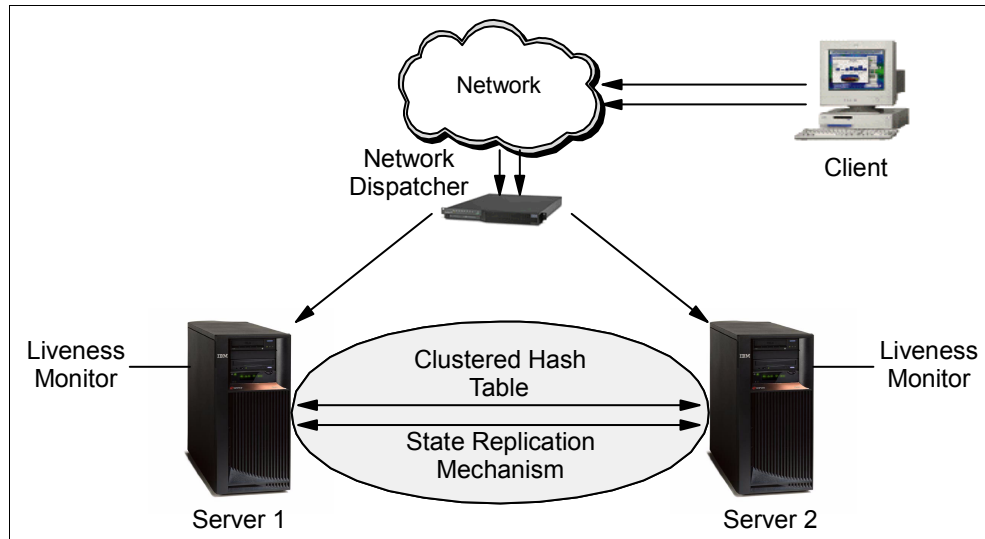


Figure 14-3 High availability: Peer model

In the event that one node fails (failover), the failed Web server traffic is routed to one of the other operational Web servers according to the configuration of the network dispatcher.

14.2 A working primary or backup with takeover IP model

This scenario provides the steps that are necessary to get a simple HA environment up and running between two iSeries servers. On each server is an identical, yet separate, instance of the HTTP Server (powered by Apache).

14.2.1 Problem definition

For this scenario, let's suppose that a customer has a network as presented in Figure 14-4. They have two iSeries servers that are available for serving a standard Web application across either a public or private network. This Web application involves the serving of standard HTML and graphics only. The primary iSeries as23 may be down for any reason:

- ▶ Planned outages such as hardware or software maintenance
- ▶ Unplanned outages such as power loss, fire, and so on

They want the ability to seamlessly move all active clients to the backup server as24. And, when the primary server as23 is brought back online, they want to move back seamlessly all the clients.

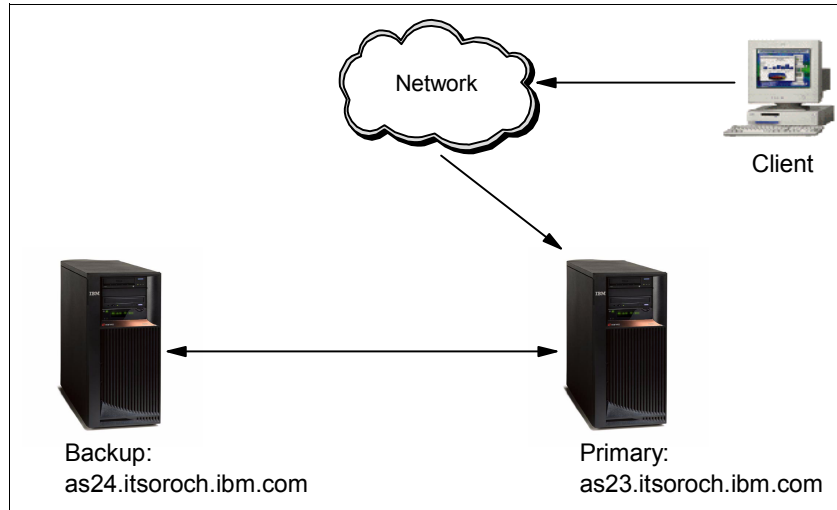


Figure 14-4 Primary or backup with IP takeover: Problem definition

14.2.2 Solution definition

The solution is to take advantage of the Highly Available HTTP Server first introduced at V5R1 for OS/400 Web applications. As shown in Figure 14-5, it takes advantage of the two separate iSeries servers as23 and as24 to provide a primary or backup with IP takeover HA solution.

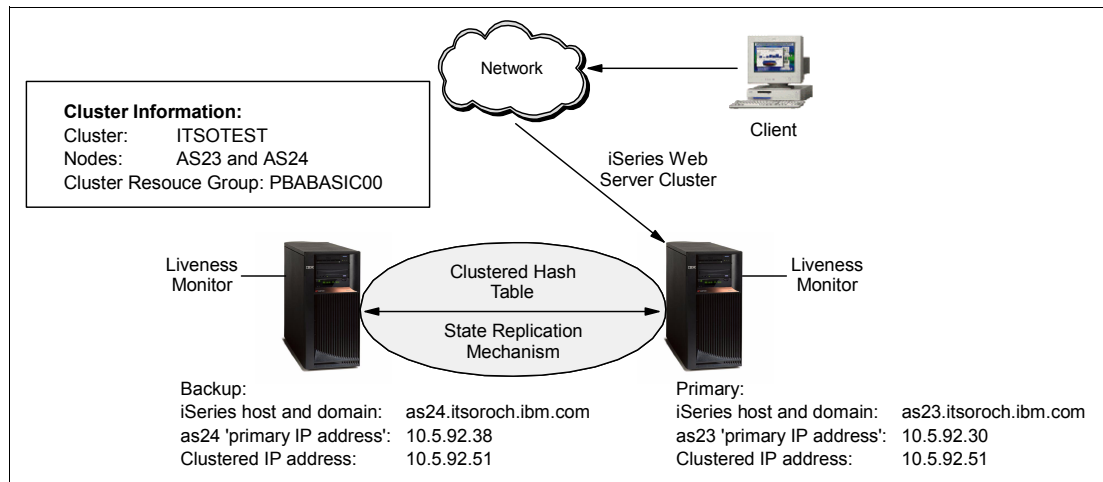


Figure 14-5 Primary or backup with IP takeover: Solution definition

14.2.3 Assumptions

The software and hardware used in this scenario has the following characteristics:

- ▶ We are using V5R2 of OS/400. This scenario can also be created for V5R1, but some of the interfaces to iSeries clustering have changed.
- ▶ If you plan to use iSeries Navigator and the Simple Cluster Management interface to configure the cluster, you must install 5722-SS1 OS/400 Option 41 (HA Switchable Resources). OS/400 Option 41 is not necessary if you plan on using the 5250 commands to configure your cluster.

- Both iSeries servers must have a routable IP address that is accessible from each other server. This is shown in Figure 14-5 that as23 can PING 10.5.92.38 and as24 can PING 10.5.92.30.

Tip: Do not confuse these IP addresses on both iSeries servers with the clustered IP address of 10.5.92.51. This clustered IP address is routable (that is, if a client on the same subnet wants to communicate to this IP address, it uses ARP to resolve the MAC address on the iSeries) but cannot be active on both systems at the same time.

14.2.4 How to

To configure the primary and backup servers in this scenario, perform the following tasks as explained in the sections that follow:

- Step 1: Validate system and TCP/IP settings on both iSeries servers
- Step 2: Creating the HA cluster for ITSOTEST for nodes AS23 and AS24
- Step 3: Adding HA clustering directives to both httpd.conf configurations
- Step 4: Testing the primary and backup servers

Note: iSeries Navigator provides a simple graphical user interface (GUI) tool to manage a cluster of iSeries servers. We chose to use a 5250 command entry in our configuration of HA clustering on the iSeries at V5R2 because the CL commands that are provided offer more features and choices not provided by the GUI. Be sure to make your own choice.

Step 1: Validate system and TCP/IP settings on both iSeries servers

Tip: In addition the steps outlined in this section, you *must* review the Information Center's resources for clustering:

<http://publib.boulder.ibm.com/html/as400/infocenter.html>

On this site, select your version and language, and click **GO!** Search for "clusters". The document that helps you the most in this step is entitled "Cluster configuration checklist".

Before you start, you need to validate and possibly modify some of the system and TCP/IP configuration settings on both iSeries servers. These instructions demonstrate what is needed on both systems by showing you what we did on as23. In your own environment, you must perform these steps for both the primary and backup servers.

1. Ensure that clustering is enabled for both iSeries servers:
 - a. Enter the Display Network Attributes (DSPNETA) command. On this Display Network Attributes display, page down until you see the value set for the Allow add to cluster as highlighted in Figure 14-6. This value should be set to *ANY on both iSeries servers.

Display Network Attributes	
	System: AS23
Allow add to cluster	*ANY
Modem country or region ID	
	Bottom
Press Enter to continue.	
F3=Exit F12=Cancel	

Figure 14-6 Primary or backup with IP takeover: Allow add to cluster should be *ANY

- b. If necessary, you can change this value using the Change Network Attributes (CHGNETA) command:

```
CHGNETA ALWADDCLU(*ANY)
```

2. Create the same routable IP address on both iSeries servers to be used as the clustered IP address. This IP address is the “well-known” IP address at which your Web application is bound to via the Listen directive. That is, a Domain Name System (DNS) server must have an A record that maps the primary host's name of as23.itsoroch.ibm.com to 10.5.92.51.

Tip: It may feel strange to create the same routable IP address on both iSeries. But, as long as both are not active at the same time this is OK. It is the IP address takeover feature of OS/400's HA clustering that automatically allows only one of the iSeries servers to have 10.5.92.51 active at one time. To be clear, you never manually make 10.5.92.51 active on either iSeries server. HA clustering's IP takeover does this for you.

Available with V5R2 of OS/400 is a new feature called *Virtual IP Address with proxy ARP* (VIPA with proxy Apache Portable Runtime (ARP)). This VIPA is routable in the same subnet as the other IP addresses associated with the physical Network Interface Cards (NIC) on your iSeries servers. VIPAs can provide extremely valuable fault tolerance for situations where you have two or more NICs configured to all be in the same subnet on the same iSeries. This scenario proceeds to create a clustered IP address on both iSeries as VIPA with proxy ARP.

The added feature of fault tolerance (for a failure in a NIC) using the VIPA addresses is detailed in *iSeries IP Networks: Dynamic!*, SG24-6718.

This scenario works equally well with new local area network (LAN) interfaces (rather than the virtual IP addresses that we use) of 10.5.92.51 on both iSeries. These can be created using 5250 commands such as Add TCP/IP Interface (ADDTCPIFC).

But, if you want HA for your Web server, you should follow up and implement VIPA with proxy ARP as part of a total solution that includes fault tolerance for a failure in one of your NICs.

- a. Using iSeries Navigator create a VIPA with proxy ARP that will be the clustered IP address on both iSeries servers. Start iSeries Navigator and connect to as23. Expand **Network** → **TCP/IP Configuration** → **IPv4**.

Note: 5250 command entry cannot be used to create VIPA with proxy ARP in V5R2. You must use iSeries Navigator.

- b. Right-click **Interfaces** and select **New Interface** → **Virtual IP**.
- c. Follow the New IPv4 Interface wizard to create the new VIPA interface. When prompted, specify the following values:
- IP address: 10.5.92.51
 - Description: VIPA with proxy ARP
 - Subnet mask: 255.255.255.255
 - Do you want to start this TCP/IP interface every time TCP/IP is started?: **No**
 - Do you want to start this TCP/IP interface now?: **No**
- d. Change the properties of this VIPA address to “turn on” the proxy ARP feature:
- i. Right-click the IP address **10.5.92.51** and select **Properties**.
 - ii. Select the **Advanced** tab and click **Enable proxy ARP**.
 - iii. Click **OK** to save your changes.

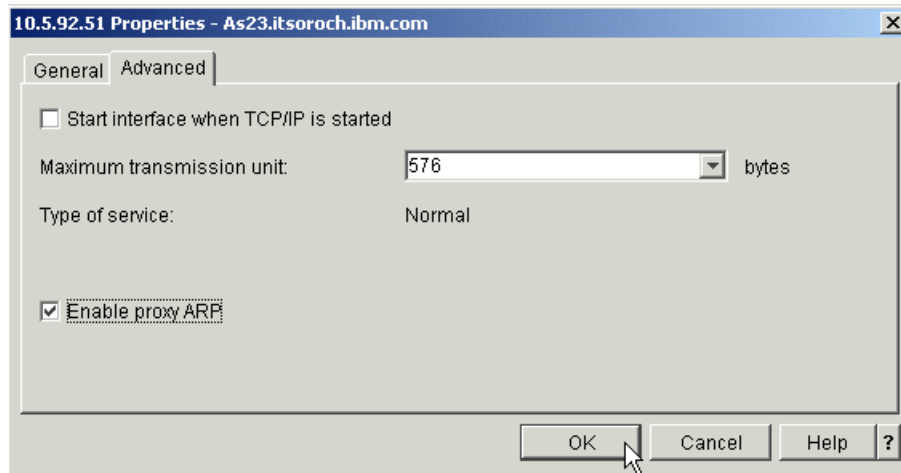


Figure 14-7 Primary or backup with IP takeover: Changing the VIPA address to enable proxy ARP

3. Make sure that the loopback IP address 127.0.0.1 is configured and started on both iSeries servers.
4. Make sure that the Internet Daemon (INETD) server is started on both iSeries servers:
 - a. Using iSeries Navigator, expand **Network** → **Servers** and then click **TCP/IP**. In the panel on the right, a list of TCP/IP servers and their status (started or stopped) is updated. Make sure the INETD server has the status of *started*.
 - b. If the INETD server has the status of *stopped*, right-click **INETD** and select **Start**.

Tip: Because INETD is needed for the proper operation of HA clustering on your iSeries servers, we recommend that you change the properties of the INETD to Start when TCP/IP is started.

5. Make sure that Liveness Monitor can run unimpeded in the QBATCH subsystem on both iSeries servers. OS/400's HA clustering uses a batch job to slow poll the primary server to determine if this system is still available to the network. By default, this batch job is started in QBATCH. You must ensure that the job queue for QBATCH is large enough to always allow this job to run.
 - a. To check the QBATCH subsystem, use the 5250 command Display Subsystem Description (DSPSBSD):
 DSPSBSD SBSD(QBATCH)
 Select option 6 (Job queue entries). As highlighted in Figure 14-8 on iSeries as23, the maximum active jobs allowed in QBATCH is 4.

Display Job Queue Entries												
										System: AS23		
Subsystem description: QBATCH				Status: ACTIVE								
Seq	Job		Max	-----Max by Priority-----								
Nbr	Queue	Library	Active	1	2	3	4	5	6	7	8	9
10	QBATCH	QGPL	4	*	*	*	*	*	*	*	*	*
20	QS36EVOKE	QGPL	*NOMAX	*	*	*	*	*	*	*	*	*
50	QTXTSRCH	QGPL	*NOMAX	*	*	*	*	*	*	*	*	*

Figure 14-8 Primary or backup with IP takeover: Max Active for QBATCH showing 4

- b. To ensure the testing will succeed, use the Change Job Queue Entry (CHGJOBQE) command to change the maximum active to no maximum:

```
CHGJOBQE SBSDBATCH JOBQ(BATCH) MAXACT(*NOMAX)
```

6. Verify that the basic TCP/IP configuration is correct.

Test the interconnectivity of all the servers and clients to ensure that the Step 2 has the best chance of succeeding. Refer to the network diagram in Figure 14-5 on page 360.

- a. Verify that primary as23 can verify TCP/IP Connection (PING) backup as24 at IP address 10.5.92.38.
- b. Verify that backup as24 can PING primary as23 at IP address 10.5.92.30.
- c. Conversely, make sure that a PING to clustered IP address 10.5.92.51 times out. That is, this IP address should not be active on any host in the 10.5.92.0 subnetwork.
- d. Verify that when the client PINGs the primary as23.itsoroch.ibm.com, the name is resolved to 10.5.92.51. However, this PING should time out.

Step 2: Creating the HA cluster for ITSOTEST for nodes AS23 and AS24

You can perform the following steps on either iSeries server as23 or as24. For the purposes of demonstration, we create the HA cluster ITSOTEST from as23.

Tip: The 5250 command GO CMDCLU provides a list of HA clustering commands that are available.

1. On iSeries server as23, use the Create Cluster (CRTCLU) command to create the cluster ITSOTEST and node AS23.

```
CRTCLU CLUSTER(ITSOTEST) NODE((AS23 ('10.5.92.30')))
```

The CRTCLU command should complete without errors.

2. To see the status of your newly created cluster ITSOTEST, enter the Display Cluster Information (DSPCLUINF) command:

```
DSPCLUINF CLUSTER(ITSOTEST)
```

As shown in Figure 14-9, your cluster is created. Currently the cluster has one node. In our case, this node is named AS23 and it has the active interface of 10.5.92.30.

Display Cluster Information					
Cluster	:	ITSOTEST		
Consistent information in cluster	:	*YES			
Current cluster version	:	3		
Current cluster modification level	:	0			
Configuration tuning level	:	*NORMAL		
Number of cluster nodes	:	1		
Detail	:	*BASIC		
Cluster Membership List					
			Potential		
Node	Status	Node	Mod	Device	
		Vers	Level	Domain	-----Interface Addresses
AS23	Active	3	0	*NONE	10.5.92.30

Figure 14-9 Primary or backup with IP takeover: Display cluster information showing one node AS23

3. Create the second cluster node AS24 by entering the Add Cluster Node Entry (ADDCLUNODE) command (on iSeries server as23):

```
ADDCLUNODE CLUSTER(ITSOTEST) NODE(AS24 ('10.5.92.38'))
```

The ADDCLUNODE command should complete without errors.
4. To see the status of your newly created node AS24 in cluster ITSOTEST, enter the DSPCLUINF command:

```
DSPCLUINF CLUSTER(ITSOTEST)
```

As shown in Figure 14-10, node AS24 is added to your cluster ITSOTEST.

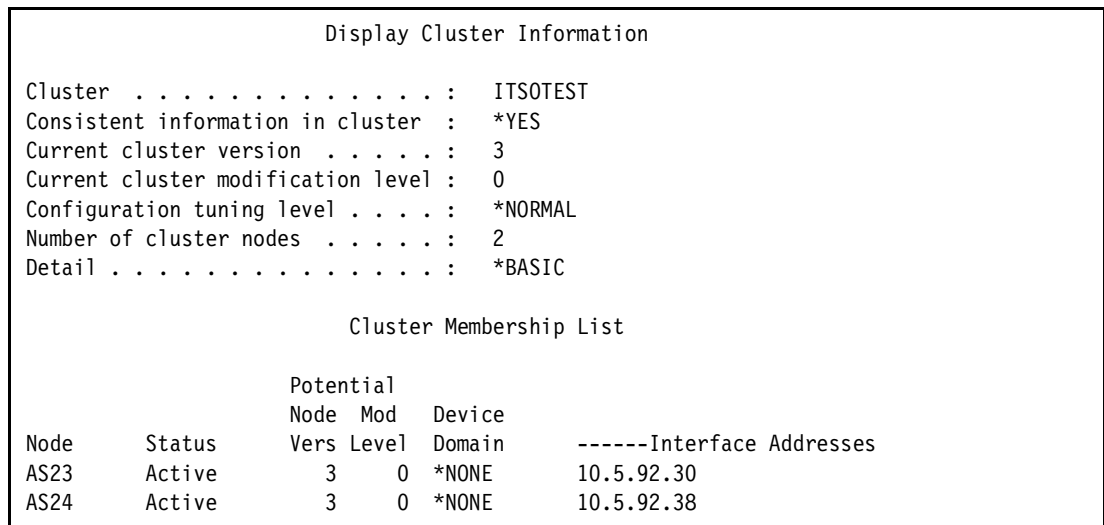


Figure 14-10 Primary or backup with IP takeover: Display cluster information two nodes: AS23, AS24

Step 3: Adding HA clustering directives to both httpd.conf configurations

Now that you established a cluster between the two iSeries servers as23 and as24, it is time to add to this cluster a CRG. You do this by creating a basic HTTP Server (powered by Apache) configuration on iSeries server as23 and then adding the necessary HA server directives. Then you make an identical copy of this HTTP Server (powered by Apache) configuration and Web site on iSeries server as24.

1. Using the administration GUI create on iSeries server as23 an HTTP Server (powered by Apache) server with the attributes shown in Table 14-1.

Tip: Make a special note of the server name and IP address.

Table 14-1 Primary or backup with IP takeover: Create New HTTP Server wizard required parameters

Create HTTP Server wizard parameter	Value
Server name	PBABASIC00 Note: This server name also becomes the name of the CRG.
Server root	/tcp52d00/basicConfig
Document root	/tcp52d00/basicConfig/ITSOco
On which IP address and TCP/IP port do you want your server to listen?	IP address: 10.5.92.51 Note: This is the address of the clustered IP address shown in Figure 14-5 on page 360. Port: 8000
Do you want your new server to use an access log?	Yes

2. Add the HA server directives to the PBABASIC00 server on as23:

- a. Select the **Manage** tab.
- b. For Server, select **PBABASIC00**. For Server area, select **Global configuration**.
- c. In the left pane, click **System Resources**.
- d. Select the **Highly Available Server** tab.
- e. Select **Enable HTTP server to be highly available**. This expands your options for this tab as shown in Figure 14-11. Specify the following options:
 - i. Select **Primary/backup with IP takeover**.
 - ii. Deselect **Enable highly available CGI programs**.
 - iii. In the Liveness monitor settings area, enter the Liveness check URL:

```
http://10.5.92.51:8000/index.html
```

This URL is slow polled (as defined by the Time between liveness checks field which is next) by the backup server to determine if the primary has failed. We recommend that this page not be a complex dynamic page but rather one that is simply HTML. Avoid HTML pages that include server-side includes (SSIs) too. Our simple Web application's index.html (home page) has static content only and makes a good liveness test.
 - iv. For the remaining parameters, keep the defaults.
- f. Click **OK**.

Highly Available Server **Denial of Service** **Advanced**

Specify one specific server IP address to listen on: ?

	IP address	Port	FRCA
Example	All IP addresses	80	Disabled
<input type="radio"/>	10.5.92.50	8000	Disabled

Add

☒ Enable HTTP server to be highly available ?

☒ Primary/backup with IP takeover
☐ Primary/backup with network dispatcher
☐ Peer

☐ Enable highly available CGI programs ?

Liveness monitor settings

Liveness check URL: ?

Time between liveness checks: or... ?

Maximum time to wait for responses: or... ?

Maximum number of recovery attempts: or... ?

OK **Apply** **Cancel**

Figure 14-11 Primary or backup with IP takeover: Enabling HA for the PBABASIC00 server on as23

3. Display the httpd.conf configuration file to view three new directives that are added as shown in Figure 14-12.

```

2    LoadModule ha_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
...
19    HAModel PrimaryBackupWithIPTakeover
20    LmURLCheck http://10.5.92.51:8000/index.html
...

```

Figure 14-12 Primary or backup with IP takeover: Three new directives to support HA in PBABASIC00

4. Create an identical HTTP Server (powered by Apache) configuration and Web application on your backup iSeries server as24. There are many ways to accomplish this. As a high-level overview, we recommend this method:
 - a. Copy and paste the entire Web application's HTML, GIFs, and other collateral from the IFS on as23 to as24. Place these files in a similar directory structure.
 - b. Use the administrative GUI to create another PBABASIC00 HTTP Server (powered by Apache) configuration on as24 using the same information found in Table 14-1.
 - c. Copy and paste the new HA directives from the PBABASIC00 httpd.conf configuration file on as23 to as24.

Tip: In the end, your objective is to have the same Web application and configuration file on both as23 and as24.

5. Since this is a test, we took the liberty to make some minor modifications to the index.html (home page) on both the primary as23 and backup as24 iSeries servers. We added the text “Welcome to Primary” on server as23 and “Welcome to Backup” on server as24. We did this so the client Web browser could see the difference when it is automatically and seamlessly switched from the primary to the backup iSeries server. Otherwise, it is difficult to tell. You can see the differences in the index.html file on both servers in Figure 14-13 and Figure 14-16 on page 370.

Step 4: Testing the primary and backup servers

To test the HA server environment, start the PBABASIC00 server on as23 and then start PBABASIC00 on as24.

1. Start the primary HTTP server PBABASIC00 on as23. Using the administrative GUI, make sure select server **PBABASIC00** on iSeries server as23 and then click **Start**. This starts the HTTP Server (powered by Apache) PBABASIC00. This also automatically starts the IP address 10.5.92.51 on iSeries server as23. The primary server is started first and the backup server is started second.

Tip: The CRG is *not* started because the backup server is still inactive. However, your Web site is up and running on the primary, you see in the next step.

2. Verify that your primary server is up and running. From a Web browser, enter the URL:
http://10.5.92.51:8000

You should see your home page as shown in Figure 14-13. The modification we made to the index.html file to say, “Welcome to Primary” to distinguish this page from the similar one on the backup server.

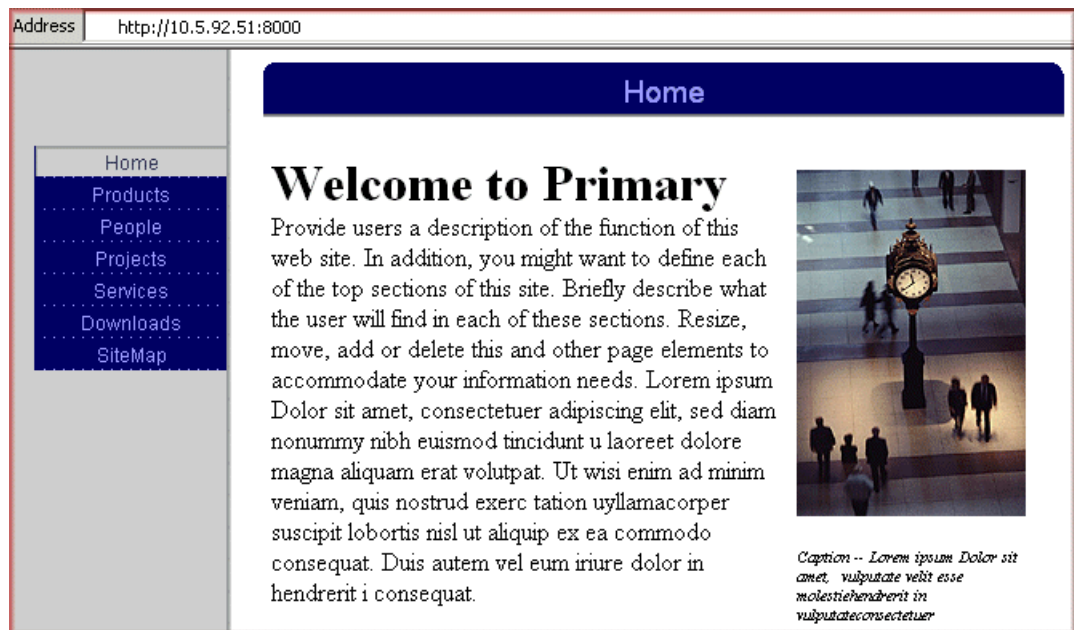


Figure 14-13 Primary or backup with IP takeover: Primary server on as23 up and running

3. Start the backup HTTP server PBABASIC00 on as24. Using the administrative GUI, make sure you select server **PBABASIC00** on iSeries server as24 and then click **Start**. This starts the HTTP Server (powered by Apache) PBABASIC00.

Tip: This does *not* start the IP address 10.5.92.51 on iSeries server as24. Only during a failure of the primary iSeries server as23 does as24 take over and make active this IP address.

4. Now that the backup server is started, this automatically creates and starts the CRG PBABASIC00. To list all the CRGs in cluster ITSOTEST, enter the Display CRG Information (DSPCRGINF) command:

```
DSPCRGINF CLUSTER(ITSOTEST) CRG(*LIST)
```

As highlighted in Figure 14-14, our CRG PBABASIC00 is active on primary node AS23.

Tip: Use the following 5250 command for even greater detail about the CRG:

```
DSPCRGINF CLUSTER(ITSOTEST) CRG(PBABASIC00)
```

Display CRG Information			
Cluster	:	ITS0TEST	
Cluster Resource Group	:	*LIST	
Consistent Information in Cluster:	:	*YES	
Number of Cluster Resource Groups:	:	2	
Cluster Resource Group List			
Cluster Resource Group	CRG Type	Status	Primary Node
PBABASIC00	Application	Active	AS23
PBABFLOARB	Data	Inactive	AS23

Figure 14-14 Primary or backup with IP takeover: CRG PBABASIC00 is active on primary node AS23

5. Use the Change CRG Primary (CHGCRGPRI) command to switch the clustered application PBABASIC00 from the primary node AS23 to the backup node AS24:

```
CHGCRGPRI CLUSTER(ITSOTEST) CRG(PBABASIC00)
```

The CHGCRGPRI command should complete without errors.

Tip: One of the major effects of this command is that iSeries server as24 does an *IP takeover* of the IP address 10.5.92.51. That is, after the CHGCRGPRI command has completed, the IP address 10.5.92.51 is inactive on server as23 and active on server as24.

- Now node AS24 is the primary server for the CRG PBABASIC00. Enter the Display CRG Information (DSPCRGINF) command:

```
DSPCRGINF CLUSTER(ITSOTEST) CRG(*LIST)
```

As highlighted in Figure 14-15, CRG PBABASIC00 is now active on (the new) primary node AS24.

Display CRG Information			
Cluster	:	ITSOTEST	
Cluster Resource Group	:	*LIST	
Consistent Information in Cluster:	:	*YES	
Number of Cluster Resource Groups:	:	2	
Cluster Resource Group List			
Cluster Resource Group	CRG Type	Status	Primary Node
PBABASIC00	Application	Active	AS24
PBABFLOARB	Data	Inactive	AS23

Figure 14-15 Primary or backup with IP takeover: CRG PBABASIC00 is active on primary node AS24

- As a final confirmation, simply refresh the Web browser which still has `http://10.5.92.51:8000` for the URL. As shown in Figure 14-16, the Web client was seamlessly moved to the backup HTTP server on iSeries as24. The only way you can really tell from the client's point of view is from the "Welcome to Backup" text we modified in the index.html home page.

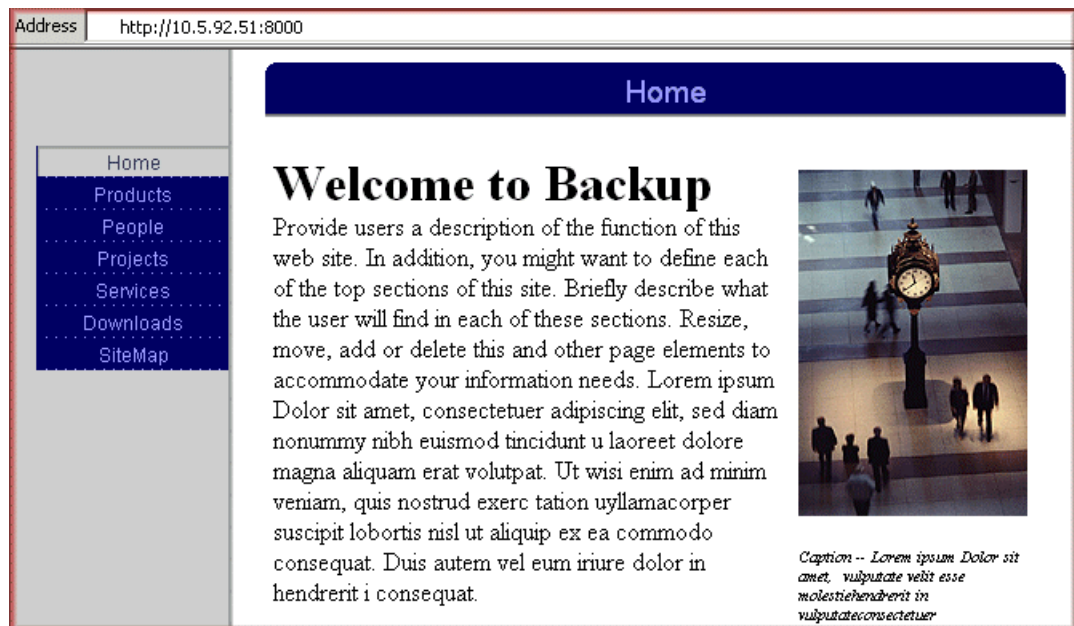


Figure 14-16 Primary or backup with IP takeover: Backup server on as24 'up and running'

The ability of the HTTP Server (powered by Apache) to take advantage of the iSeries HA APIs is a powerful enhancement to the Apache server. This is another demonstration of the integration that the Rochester lab has done to bring the Apache server to the iSeries server.

14.3 For more information

Highly available HTTP servers and iSeries clustering resources include:

- ▶ *HTTP Server: What's new*, which includes information about the 17 December 2001 Highly Available HTTP Server announcement

<http://www.ibm.com/servers/eserver/iseries/software/http/news/sitenews.html>

- ▶ Documentation Center information about highly available Web servers

<http://www-1.ibm.com/servers/eserver/iseries/software/http/docs/doc.htm>

On this site, search for “high availability”.

- ▶ High Availability and Clusters home page

<http://www.ibm.com/servers/eserver/iseries/ha/>

- ▶ Clustering documentation in the iSeries Information Center

<http://publib.boulder.ibm.com/html/as400/infocenter.html>

On this site, select your version and language, and click **GO!** Then search for “clusters”.

- ▶ The IBM Redbook *Clustering and IASPs for Higher Availability on the IBM @server iSeries Server*, SG24-5194



National language considerations

This chapter discusses national language considerations on the iSeries server as they relate to the HTTP Server (powered by Apache). It also discusses many associated applications such as the Digital Certificate Manager (DCM) and the Cryptographic Service Provider, for example. It provides information for you to view these applications in many of the world's languages.

Tip: It is interesting to note that some of the applications found on the iSeries Tasks page have their graphical user interface (GUI) driven by Net.Data (for example the DCM). Others have their GUI driven by servlet (for example the administration GUI for HTTP Server (powered by Apache)). Because of this underlying difference, each handles national language support (NLS) in a different manner!

15.1 Installing secondary languages

The GUI used to configure the HTTP Server (powered by Apache) can be installed in many different languages. To use different languages, you must install the licensed product:

- ▶ 5722-DG1: If you need different languages for the administration GUI
- ▶ 5722-SS1, option *BASE and 3: For the initial iSeries TASK page
- ▶ 5722-SS1, option 34: For the Digital Certificate Manager (DCM)
- ▶ 5722-AC3: For the Cryptographic Service Provider
- ▶ 5722-JV1 Developer Kit for Java

Before we can work with different languages, we have to check if the licensed program is installed and if it is installed in the needed language. To check this, follow these steps:

1. Enter the OS/400 command GO LICPGM and press Page Down.
2. You should now see the Work with Licensed Programs display (Figure 15-1). From here choose option 20 (Display installed secondary languages) to check if any secondary languages are installed.

LICPGM	Work with Licensed Programs	System: AS20
Select one of the following:		
Secondary Languages		
20. Display installed secondary languages		
21. Install secondary languages		
22. Delete secondary languages		
Redistribution		
40. Create distribution media		
41. Work with installation profiles		
Completion Status		
50. Display log for messages		
		More...
Selection or command		
===>		
F3=Exit F4=Prompt F9=Retrieve F12=Cancel F13=Information Assistant		
F16=AS/400 Main menu		

Figure 15-1 Work with Licensed Programs: Page down

3. You now see the Display Installed Secondary Languages display (Figure 15-2). In our case, 2924 (English) is the primary, and 2929 (German) is the secondary language. Enter 5 (Display installed Licensed Program) to see the installed licensed programs associated with that secondary language.

Tip: You can find a national language feature code matching table on the Web at:

<http://publib.boulder.ibm.com/infocenter/iseres/v5r3/ic2924/index.htm?info/rzahc/rzahcnlvfeaturecodes.htm>

Display Installed Secondary Languages			System:	AS20
Primary language	:	2924		
Description	:	English		
Type options, press Enter.				
5=Display installed Licensed Program				
Option	Language	Description		
5	2929	German		

Figure 15-2 Display Installed Secondary Languages

4. In the Display Installed Secondary Language Licensed Programs display (Figure 15-3), verify the Installed Status of the products. If any of these is in status *ERROR, try to re-install the product or contact your local service representative.

Display Installed Secondary Language Licensed Programs			System:	AS20
Secondary language	:	2929		
Description	:	German		
Licensed Program	Installed Status	Description		
5722SS1	*COMPATIBLE	OS/400 - Digital Certificate Manager		
5722DG1	*COMPATIBLE	IBM HTTP-Server		
5722DG1	*COMPATIBLE	Triggered Cache Manager		

Figure 15-3 Display Installed Secondary Languages: Option 5 (LPPs)

If no secondary language is installed on your system, refer to iSeries Information Center or to Chapter 10 in *Software Installation V5R2*, SC41-5120.

15.2 Net.Data based: iSeries Tasks page and DCM

Both, the iSeries Tasks page and the DCM are implemented as a Net.Data application. Net.Data based GUIs use iSeries host settings to determine which language they should present to the client. Therefore, you must check the language settings in the user profile that you used to sign on to the iSeries Tasks page, to display the correct language. As shown in Figure 15-4, use the OS/400 command Display User Profile (DSPUSRPRF) USRPRF(WOLFGANGP) to check the settings for your OS/400 user profile.

Display User Profile - Basic	
User profile	WOLFGANGP
Previous sign-on	05/07/03 10:11:16
Sign-on attempts not valid	0
Status	*ENABLED
Date password last changed	04/29/03
Password expiration interval	*SYSVAL
Date password expires	11/01/03
Set password to expired	*NO
.	
.	
.	
Language identifier.	*SYSVAL

Figure 15-4 Display User Profile: Language identifier

If the language ID is set to *SYSVAL (which is the default), the OS/400 system value QLANGID is used to identify the desired language. To display the system value, use the OS/400 command Display System Value (DSPSYSVAL SYSVAL(QLANGID)). You should see the Display System Value display as shown in Figure 15-5.

Display System Value	
System value	QLANGID
Description	Language identifier
Language identifier	ENU Language abbreviation

Figure 15-5 Display System Value: QLANGID

In our case, this display shows ENU which means US - English. If you have some users that need to work with other languages (for example German), you can change the Language identifier parameter to DEU. The next time the user connects to any site pages served by Net.Data on this iSeries server, they see them in German.

Tip: It is important to learn that if the Web pages are served by Net.Data, the iSeries server examines the language settings of your browser. It only uses the information found in either your OS/400 user profile or the default system value. For more information about using the browser's language settings to influence the national language chosen by the iSeries, see the following section.

15.3 Servlet based: Administration GUI

The administrative GUI can be served in any language that is installed on your iSeries server.

Tip: You have to restart the Admin instance after installing the secondary languages to take advantage of the change.

How the language recognition works

The HTTP Server (powered by Apache) (and we are focusing on the administrative GUI in this section) determines which language it has to serve based on the *Accept-Language request header*. This is determined on the first request that is done from the browser to the server. All future requests from the same browser session to the HTTP Server (powered by Apache) are served in this language. If you change the language settings in your Web browser in the middle of a session, you have to restart it to see the new language.

The configuration in your browser is simple.

For Microsoft Internet Explorer

Follow these steps to configure the client's language choices:

1. Click **Start** → **Settings** → **Control Panel** → **Internet Options**.
2. The Internet Properties window (Figure 15-6) opens. Click **Languages**.

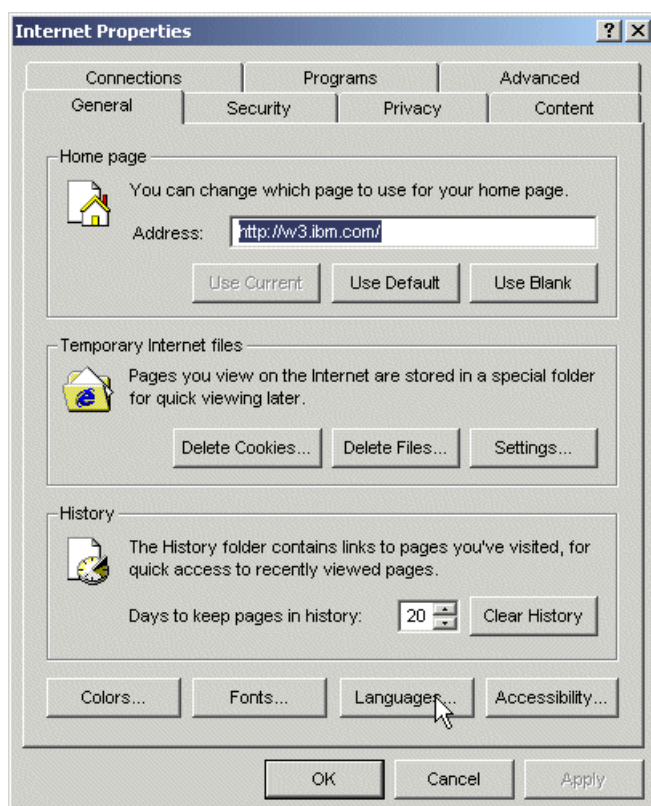


Figure 15-6 Microsoft Internet Explorer Internet Properties window

3. In the Language Preference window (Figure 15-7), click **Add**.

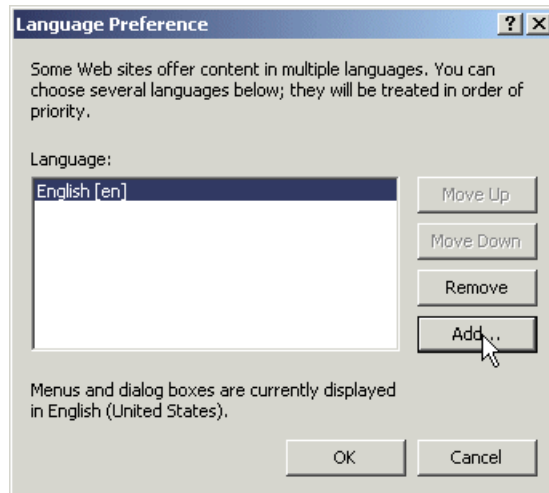


Figure 15-7 Language Preference window

4. In the Add Language window (Figure 15-8), choose the appropriate language. In our case we chose **German**. Click **OK**.

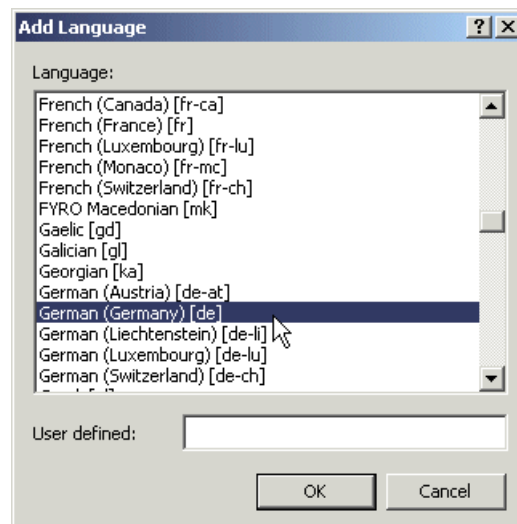


Figure 15-8 Add Language window

5. Click **OK** on the Language Preference window (Figure 15-7) and on the Internet Properties window (Figure 15-6) to save all your changes.

For Netscape Navigator

Follow these steps to configure the client's language choices:

1. From the menu bar, click **Edit** → **Preferences** as shown in Figure 15-9.

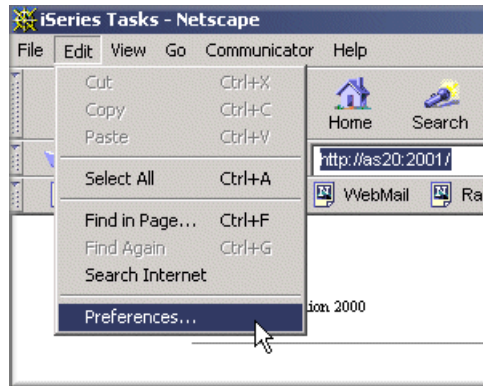


Figure 15-9 Using Netscape Navigator to select a language

2. In the Preferences window (Figure 15-10), under Category, expand **Navigator** and select **Languages** to see what language the browser is configured to request.
3. To add a new language, click **Add** and select the appropriate one.

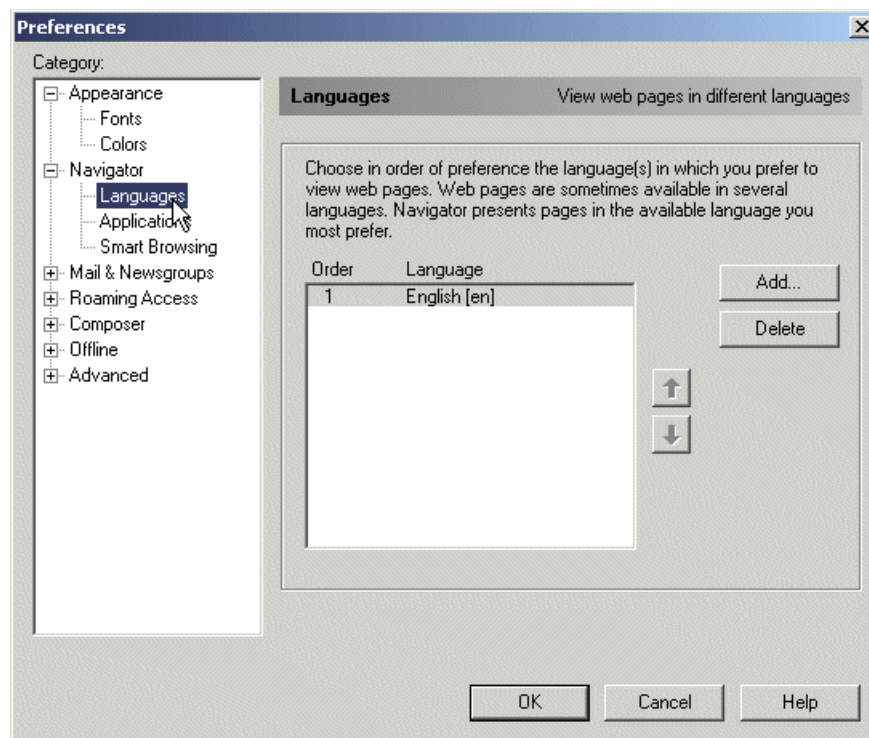


Figure 15-10 Preferences window in Netscape Navigator

4. In the Add Languages window (Figure 15-11), select the language or languages you need and click **OK**.



Figure 15-11 Add Languages window

5. Click **OK** on the Preferences window.

After you complete these steps, restart your browser. The next request to the administrative GUI (and only this interface) appears in the correct language.

Attention: Use care in regard to the sort order of the configured languages. The browser initially requests the first language in the list and steps through all configured ones. If none of the configured languages matches the ones that the HTTP Server (powered by Apache) can serve, it reverts to the LANGID parameter of the authenticated user profile. This kind of language selection, for example, is used by the IBM Web Administration for iSeries interface.

15.4 Other programs linked from iSeries Task page

Many other licensed programs use the iSeries Tasks page as their initial enter page. It is also possible to display those pages in a different language.

15.4.1 Internet Printing Protocol server for the iSeries server

This licensed product is included in 5722-SS1, option 3 (OS/400 - Extended Base Directory Support). If you want to use different languages, you have to verify that it is also installed as a secondary language.

The language recognition is the same as for the administration GUI (see 15.3, “Servlet based: Administration GUI” on page 376). If the product is installed in the needed language, you only have to check your browser’s setup and the correct language.

For more information about the Internet Printing Protocol (IPP) GUI, see IBM Publication *IBM @server iSeries Printing VI: Delivering the Output of e-business*, SG24-6250.

15.4.2 WebSphere family

Refer to the following documentation on the WebSphere on iSeries home page at:

<http://www.ibm.com/eserver/iseries/software/websphere>

15.4.3 4758 Cryptographic Coprocessor

The GUI to set up the 4758 Cryptographic Coprocessor is only translated in the following languages:

- ▶ Brazilian Portuguese
- ▶ Italian
- ▶ Japanese
- ▶ Korean
- ▶ Simplified Chinese
- ▶ Spanish
- ▶ Swiss Italian

If you do not have installed any of the above languages, the GUI is always displayed in English (if 5722-AC3 is installed in English (2924) on your system either as the primary or as secondary language). It uses the same mechanism as in 15.2, “Net.Data based: iSeries Tasks page and DCM” on page 375, to display the GUI in one of the previous languages.

15.5 Serving your own Web site in the world’s languages

To serve your own language-based sites, you do not have to install any secondary language or additional software on your iSeries server. The choice of language that the HTTP Server (powered by Apache) serves is mainly based on the Accept-Language request header, so it depends on your browsers language settings, your HTTP Server configuration, and the Web application that you are serving.

Follow these steps to prepare your HTTP Server (powered by Apache) to serve the correct language based page:

1. As shown in Figure 15-12, for Server, select your server. For Server area, select **Global configuration**.

Tip: You can configure these settings for your server’s global configuration or for an individual container.

2. In the left pane, under Server Properties, select **Content Settings**.
3. Select the **MIME** tab.

MIME: MIME stands for Multipurpose Internet Mail Extensions. It refers to an official Internet standard that specifies how messages must be formatted so that they can be exchanged between different systems.

4. On the MIME page (Figure 15-12), complete these steps:
 - a. Under Specify individual Meta (MIME) information for file extensions, click **Add**. You add one row for each national language that you will serve from this HTTP Server (powered by Apache).

Add all the file extensions you want to serve. Select **Content-language** as the type of content encoding. Both the Type and Value lists show all the possibilities. In this section, you associate a file extension to the language requested from the browser in the Accept-Language request header.

That is, a browser that is setup to request [en] encoded pages asks the HTTP Server (powered by Apache) to search for files that contain the configured file extension in the file name. In this case, for an incoming request with the Uniform Resource Identifier (URI) of /index, a search is made for the file /index.en.html or /index.html.en.

- b. When your are done adding all the languages you will be serving, click **Apply**.

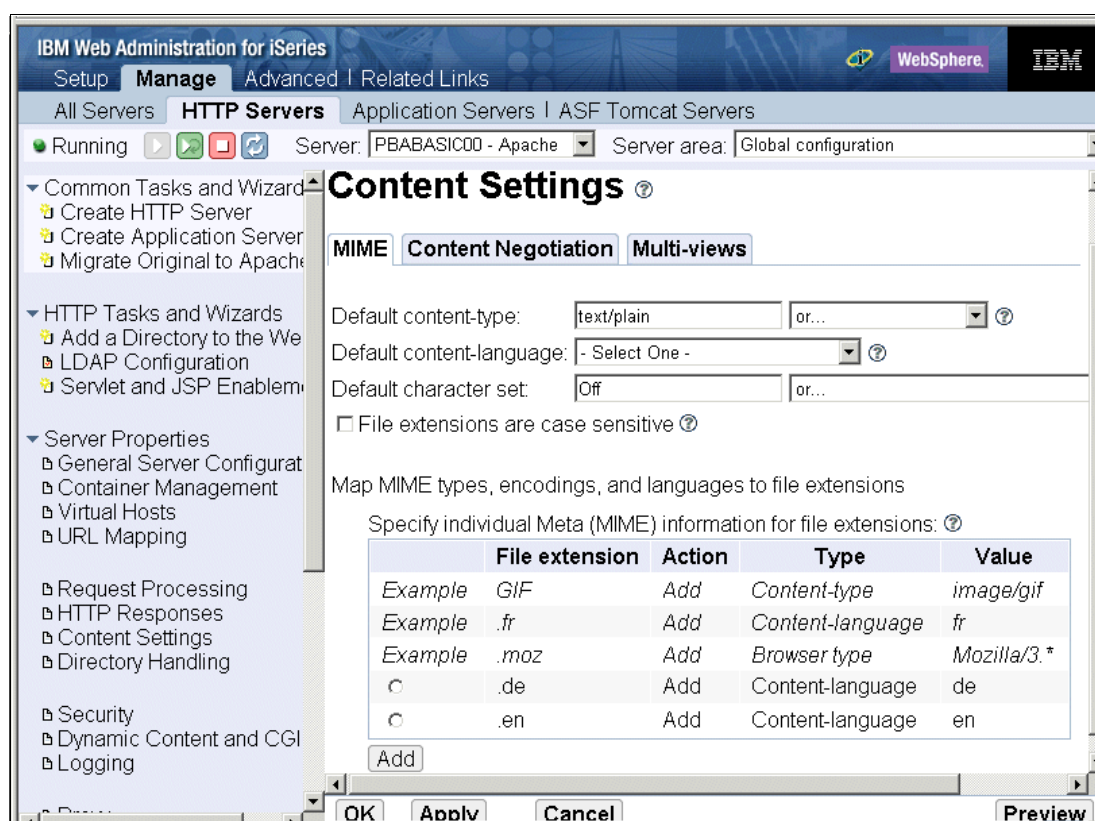


Figure 15-12 Serving language-based pages: Content Settings

5. Select the **Multi-views** tab as shown in Figure 15-13.
6. On the Multi-views page, you enable the server to honor client request options, such as the language setting. Enable the HTTP Server (powered by Apache) to search the files in the IFS that correlates to the requested language.

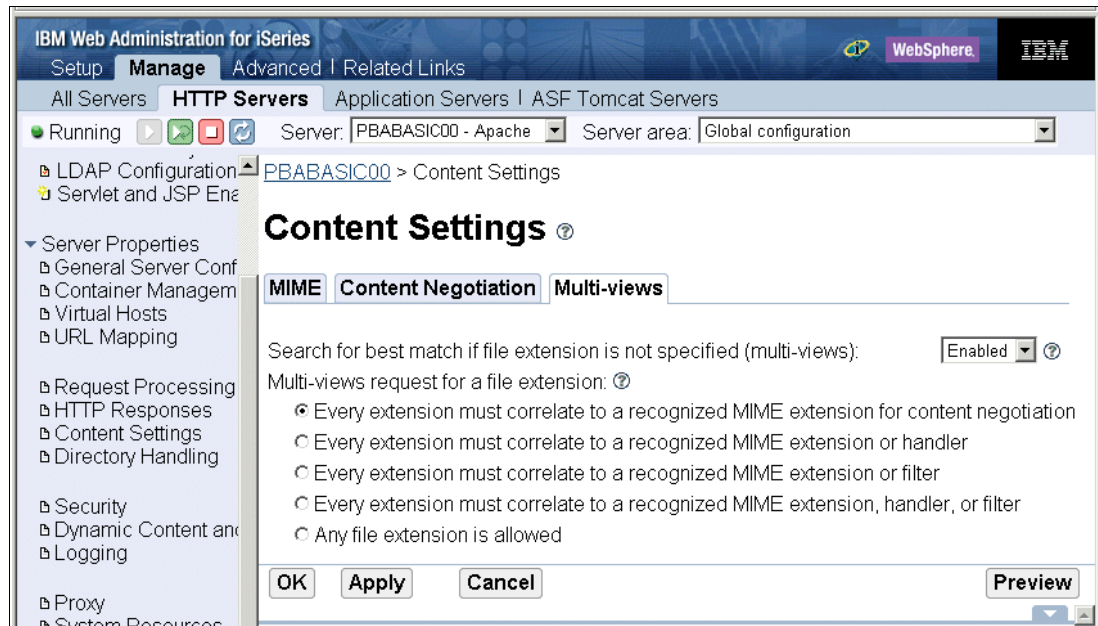


Figure 15-13 Serving language-based pages: Enabling multi-views

7. It is possible for the client's Web browser to request a language that you are not prepared to serve. To prevent the client from seeing the HTTP error 406 - Not acceptable, configure the HTTP Server (powered by Apache) to serve a default language.

As shown in Figure 15-14, select the **Content Negotiation** tab.

8. On the Content Negotiation page, click **Add** to add a default language. For Force language priority, select **Prefer/Fallback**.

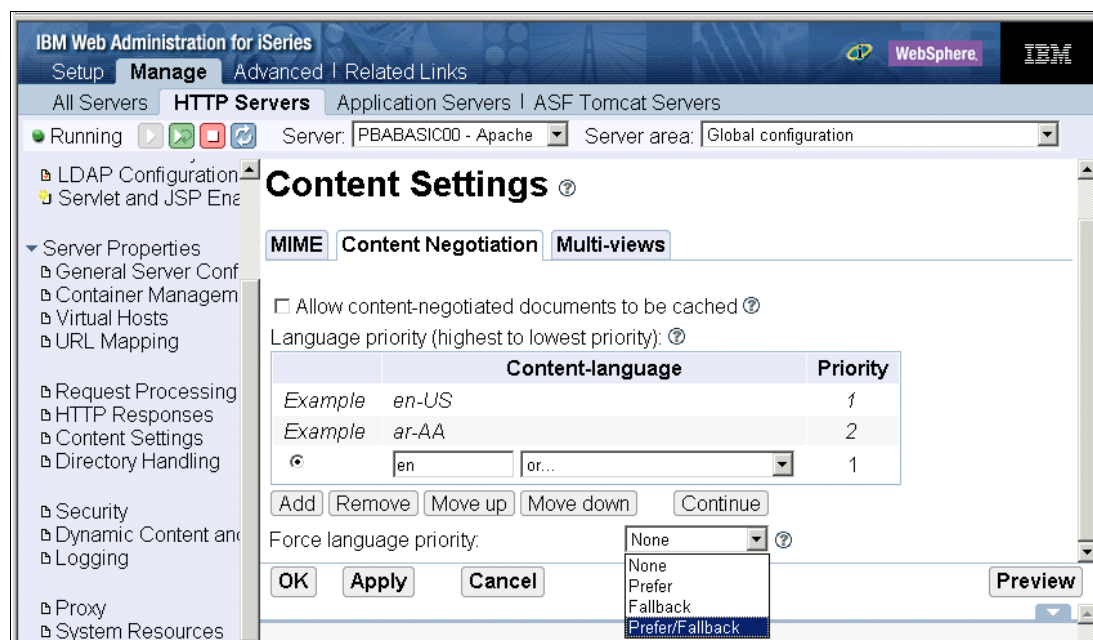


Figure 15-14 Serving language-based pages: Default language

Figure 15-15 shows the configuration directives we changed or added.

```
Options +MultiViews
AddLanguage de .de
AddLanguage en .en
LanguagePriority en
ForceLanguagePriority Prefer Fallback
```

Figure 15-15 Serving language-based pages: Directive changes

After you apply all configuration changes, restart your server.

If you are using welcome pages, you must take care of your HTML files. Due to the setup and renaming of the files, they are not named index.html anymore. They are named index.en.html or index.html.en. When the HTTP Server (powered by Apache) searches for those files and it is configured to display the welcome page index.html, it only finds the appropriate one when the filename is index.html.en. In this case, HTTP error 403 or the directory listing is sent back to the client. To prevent this, set the directive `DirectoryIndex` to *index*.



Part 4

Appendixes

To complete this IBM Redbook, we include appendixes on porting different open source projects to the iSeries server. An advantage of using the Apache server is that you do not have to wait for the developers in Rochester, Minnesota, to provide you with new features or functions. You can do it yourself.

In addition, all the examples provided in this redbook can be downloaded from the Web, allowing you to reduce the time in your transition from reading to understanding and implementation. See Appendix D, “Additional material” on page 421.



Bringing PHP to your iSeries server

Hypertext Preprocessor (PHP) is a powerful server-side scripting language for the Apache Web server. PHP is popular for its ability to process database information and create dynamic Web pages. *Server-side* refers to the fact that PHP language statements, which are included directly in your Hypertext Markup Language (HTML), are processed by the Web server. *Scripting language* means that PHP is not compiled. Since the results of processing PHP language statements is standard HTML, PHP-generated Web pages are quick to display and are compatible with most all Web browsers and platforms. PHP is for the open source Apache community as Net.Data is for the IBM community.

Restriction: PHP is not supported on the iSeries server by IBM. We provide these instructions for you to download a public domain open-source copy of a PHP engine to allow you to run PHP scripts on the iSeries server.

Specifically, IBM does *not* support:

- ▶ The open-source CGI based PHP engine
- ▶ Any of the PHP scripts that you would write against this PHP engine
- ▶ The other open source tools described in this IBM Redbook used for building the PHP engine

IBM fully supports:

- ▶ 5722-SS1 Option 33 OS/400: Portable Application Solutions Environment (OS/400 PASE) and all the utilities supplied with it
- ▶ The VisualAge® C++ compilers
- ▶ The HTTP Server (powered by Apache) support for OS/400 PASE Common Gateway Interfaces (CGIs)

To “run” PHP scripts with your HTTP Server (powered by Apache), a PHP engine is required on your iSeries server. The PHP engine is an open source product. This chapter explains how to download, compile, install, and then configure PHP on your iSeries. It explains how to install versions 4.3.0 and the older version 4.2.2 of PHP.

The PHP engine is available both as an Apache module and a CGI. Support for PHP as a module is not yet available for OS/400. The step-by-step implementation discussed in this chapter involves the CGI version of PHP running in OS/400 PASE. For a general discussion on the CGI support with the HTTP Server (powered by Apache), see 7.2, “Everything dynamic with CGI support” on page 160. This allows you to run AIX binaries directly on an iSeries. It includes the necessary patches for the minor modifications needed to the PHP source code.

Note: If you want to know why this is so great, see the article “Programming with PHP on the iSeries” for iSeries Network by David Larson and Tim Massaro. You can find this article (requires login) on the Web at:

http://www.iseriesnetwork.com/resources/artarchive/index.cfm?fuseaction=viewarticle&CO_ContentID=15746&channel=art&PageView=Search

With permission from iSeries Network, we include the article in this IBM Redbook. To skip the article, go to “Prerequisites” on page 398.

Programming with PHP on the iSeries server

Hypertext Preprocessor Language is a powerful, server-side scripting language for Web page creation. Scripting language means PHP requires no compilation, much like Perl or Rexx. Because PHP is a server-side language, you can include it directly in HTML, and it is recognized and processed by a Web server.

The first “P” in PHP is a remnant from the original acronym for Personalized Home Page. This was the term that PHP creator Rasmus Lerdorf used when he first used a set of Perl scripts to monitor access to his online resume. Since then, however, PHP has become the most popular optional module configured on Web servers. See the following Web sites:

- ▶ <http://www.netcraft.com/survey>
- ▶ http://www.securityspace.com/s_survey/data/man.200304/apachemods.html

This section introduces the PHP language and explains how to configure PHP to access DB2 Universal Database (UDB) from your Apache Web server. Then, you see examples of how iSeries shops can use PHP to create dynamic Web pages based on new or existing iSeries DB2 UDB databases.

What PHP is

PHP code can easily access database files and output HTML, resulting in non-static, up-to-date Web pages. It's a technique similar to JavaServer Pages (JSPs) or CGI binary (often called CGI-BIN) programming. Also, PHP is an open-source project. Open-source code can be useful if you want to tweak the behavior of PHP, but it's even more valuable because there are many open-source PHP applications and code samples available on the Web. This means you can get a new PHP Web project up and running quickly with little investment.

Hundreds of ready-made applications written in PHP are available as shareware, and many commercial products employ it. Until recently, PHP enjoyed a reputation for reliability and security. See “Beware of PHP bugs” on page 397.

Figure A-1 shows the difference between standard static Web pages and dynamic Web pages using server-side PHP processing. In the first scenario on the left, a standard URL request arrives at the Web server asking for Web page:

`http://www.example.com/index.html`

The Web server sees this request and returns the HTML that is in the file `somepage.html`.

Still looking at Figure A-1 with the second scenario on the right, the `index.php` file contains the special `<?php>` tag that tells the Web server to process embedded PHP statements. After PHP processes those statements, it returns HTML statements to the Web server. Those statements are then sent back to the user included in the original HTML found in the file `index.php`. Because the PHP statements can run a command or Structured Query Language (SQL) statement, we say that the Web page is *dynamically generated*, as opposed to our previous static HTML page.

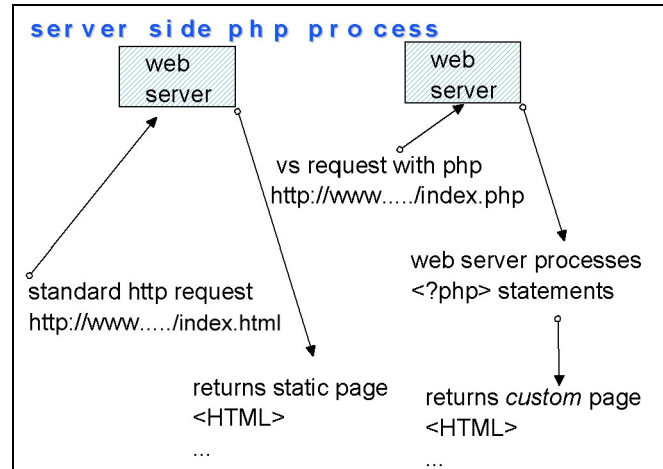


Figure A-1 Left: Standard request for a Web page
Right: Request with PHP

Why PHP

Besides the fact that PHP is so popular, why would you want to use it? There are several reasons:

- ▶ **Easy to use:** As mentioned earlier, PHP is a scripting language included directly in HTML. This means that getting started is easy. There's no need to compile PHP programs or spend time learning tools that create PHP. You can simply insert statements and get quick turnaround as you make changes.
- ▶ **Fully functional:** The PHP language has built-in functions to access your favorite database. With PHP, your HTML pages can reflect current information by querying those databases, or you can use information about the user viewing your HTML Web page to customize the page specifically for that user. In addition to good relational database support, PHP is a complete language that includes powerful functions. You can create classes for object-oriented programming and use flat file or Lightweight Directory Access Protocol (LDAP) databases. Plus, it includes a spell checker, Extensible Markup Language (XML) functions, image generation functions, and more.
- ▶ **Compatible and quick:** Because PHP generates plain HTML, it's compatible with all Web browsers and refreshes quickly.
- ▶ **Secure:** Although PHP is open source, it's a secure environment. One of its advantages (over, JavaScript, for example) is that all that Web clients see is pure HTML. Because the logic of the PHP program is never exposed to the client, security exposures are reduced.
- ▶ **Open source:** Another reason to use PHP is because it's an open-source project, which makes it easy to find examples and get started quickly. Here are two examples of Web sites that offer a place where PHP scripts are shared:

- <http://www.sourceforge.net>
- <http://www.phpresourceindex.com>

A code example

Example A-1 shows us a simple "HelloWorld!" example. This is as simple as it gets. The file starts as a normal HTML file. We simply insert PHP statements following the <?PHP tag. The <?PHP tag is the signal to the HTML processor that PHP processing is necessary. The print statement is a PHP statement.

Example: A-1 HelloWorld PHP example

```
<html>
<head><title>Standard HTML Page with PHP
HelloWorld</title>
<body>
<?PHP
    print "Hello World";
    print "<br> Generated with <b>PHP</b>";
?>
</body>
</html>
```

To make this example a little more useful, we add the statements shown in Example A-2 to query the state of our Web server.

Example: A-2 Changes made to HelloWorld

```
<?PHP
    print "Hello World from System:" .
        $HTTP_SERVER_VARS['HTTP_HOST'];
    print phpinfo();
?>
```

The result of our PHP program is similar to what is shown in Figure A-2. This is a dynamic Web page that contains the name of our Web server and a table built by PHP with details about how PHP is configured on our Web server. This is accomplished by using one of several predefined PHP variables (for example HTTP_HOST) and the PHP function **phpinfo**.

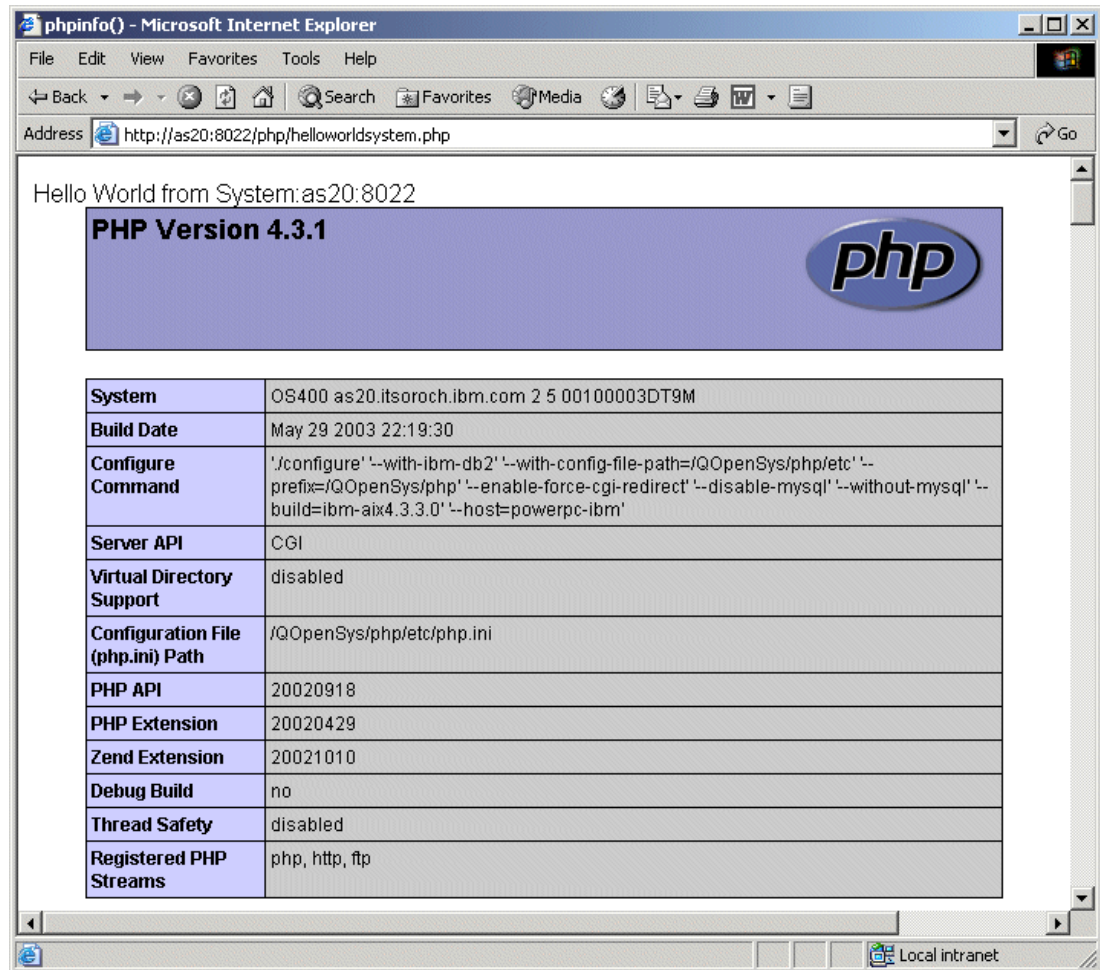


Figure A-2 Dynamic Web page generated by the PHP 'Hello World' script

PHP on the iSeries server

An iSeries user has two options to set up PHP. You can use PHP with OS/400 PASE and the HTTP Server (powered by Apache). Or you can install a Linux logical partition (LPAR) and run Apache and PHP in that partition. Table A-1 shows factors to consider before you make this decision.

Table A-1 Which is for you? PHP as a CGI in OS/400 PASE versus PHP in a Linux LPAR

Factors to consider	PHP in OS/400 PASE and Apache	PHP in Linux LPAR
OS/400 requirements	You should have V5R1 or newer. This should work for V4R5, but we have not tried it ourselves.	You must have V5R1 or newer with specific hardware to run Linux.
Cost	A cost is associated with OS/400 PASE (becomes free in V5R2). In V5R1 and prior releases, OS/400 PASE was a nominal fee of around \$100 US.	A cost is associated with Linux distribution.

Factors to consider	PHP in OS/400 PASE and Apache	PHP in Linux LPAR
Setup required	No setup is required to use PASE.	Some setup associated with the creation of a Linux partition, user IDs, and so on, and extra LPAR requires some dedicated processor resource.
Availability and compatibility	You must obtain PHP from these instructions and have AIX skills to compile PHP as new versions come out.	Linux is most compatible with new versions of PHP as they are released.
mySQL	mySQL is unavailable in PASE by default. You must download and compile it if desired.	mySQL is available as an alternative database (it is fairly common to use mySQL with PHP applications).
Web server module	PHP cannot be a Web server module. It must be a CGI process only. This matters only in extremely performance-critical Web sites.	PHP can be installed as an Apache module.
Database	An Open Database Connectivity (ODBC) driver is not necessary.	To use iSeries DB2 UDB, you must download, install, and configure iSeries ODBC Driver for Linux. This UNIX-based ODBC is free from IBM. It uses sockets to communicate between the Linux LPAR and the iSeries LPAR.

If you plan to install PHP on an iSeries server, you need to be at V5R1 or later. As mentioned in Table A-1, this can work for V4R5, but we have not tried it ourselves. You must also have installed OS/400 PASE. PASE is the AIX runtime support for iSeries. See “Prerequisites” on page 398 to see if you have the requirements for running PASE on your AS/400 or iSeries hardware.

If you plan to install PHP on a Linux LPAR, PHP is most likely included with your Linux distribution. If it is not included, the installation instructions are virtually identical to those found in the PHP distribution itself and in the PHP site frequently asked questions (FAQs) at:

<http://cvs.php.net/cvs.php?login=1>

Regardless of where you install PHP in OS/400, the configuration is the same. For the Apache Web server to recognize PHP files, you must change the Web server configuration file to recognize script aliases and allow access to the directory in which the PHP CGI executes. See Example A-3. The directory where PHP is installed may differ.

Example: A-3 Script aliases for PHP

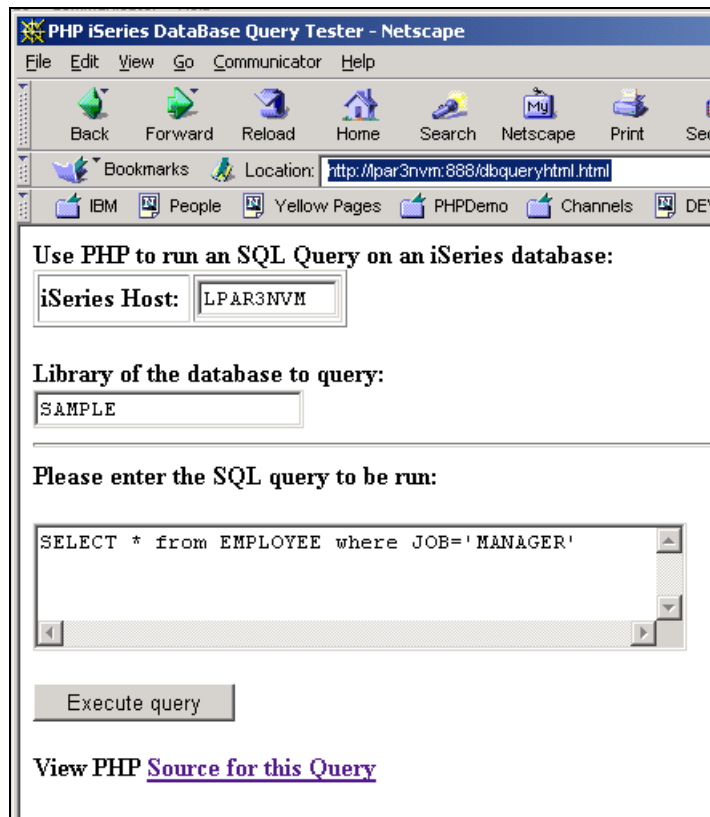
```
ScriptAlias /php-bin/ /usr/local/php/bin
AddType application/x-httpd-php .php
Action application/x-httpd-php /php-bin/php
<Directory /QOpenSys/php/bin>
    Options +ExecCGI
    order allow,deny
    allow from all
</Directory>
```

PHP as a CGI program

The next example shows a traditional HTML form that uses the Action tag to invoke a CGI program when a user clicks the Submit button. In this example, the CGI program is actually a PHP program that processes the fields in the HTML form and uses that information to query a DB2 database.

The database we use is called SAMPLE. SAMPLE is actually shipped with V5R1. To create it, follow the instructions in “Creating a sample database” on page 405.

Figure A-3 shows the basic HTML form that we use to perform a database query. Our system name is LPAR3NVM.



The screenshot shows a Netscape browser window titled "PHP iSeries DataBase Query Tester - Netscape". The address bar displays "http://lpar3nvm:888/dbqueryhtml.html". The form contains the following elements:

- Use PHP to run an SQL Query on an iSeries database:**
- iSeries Host:** A text input field containing "LPAR3NVM".
- Library of the database to query:** A text input field containing "SAMPLE".
- Please enter the SQL query to be run:** A text area containing the SQL query: `SELECT * from EMPLOYEE where JOB='MANAGER'`.
- Execute query**: A button to submit the query.
- View PHP [Source for this Query](#)**: A link to view the source code.

Figure A-3 Basic HTML form used to perform a database query

Figure A-4 shows the results of our query. Each record returned was placed in a table row.

PHP DB Query Tester - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Shop Stop

Bookmarks Location: http://mpar3nvm:888/dbqueryphp.php

IBM People Yellow Pages PHPDemo Channels DEV/2000 Home P Rochester, MN H Google IBM Linux Techn

Query performed: **SELECT * from EMP where JOB='MANAGER'**

Results:

EMPNO	FIRSTNAME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE	JOB	EDLEVEL	SEX	BIRTHDATE	SALARY
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250.00
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250.00
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175.00
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250.00
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170.00
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750.00
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150.00

New query

View PHP [Source for this Query](#)

Figure A-4 The result of the query

Example A-4 shows the dbqueryphp.php script where the actual work is done.

Example: A-4 The dbqueryphp.php script

```
<HTML>
<HEAD>
<TITLE>PHP DB Query Tester </TITLE>
</HEAD>
<BODY>
<!--dbqueryphp.php -->
<!--Called by dbqueryhtml.html -connect to sample db2 database and run an SQL
statement -->
<?php
    $host = $_POST['host'];
    $database = $_POST['database'];
    $query = $_POST['query'];
    if ($host && $database && $query) {
        $link =odbc_connect($host, "", "");
        if(!odbc_setoption($link,1,SQL_ATTR_COMMIT,SQL_TXN_NO_COMMIT)){
            echo "ERROR:unable to turn off commitment control!\n";
        }
        if(!odbc_setoption($link,1,SQL_ATTR_DBC_DEFAULT_LIB,$database)){
            echo "ERROR:unable to set default library to $database!\n";
        }
        $querynoslash =stripSlashes($query);
        $result =odbc_exec($link,$querynoslash);
    }
    ?>
    Query performed:<B><?php echo ($querynoslash);?></B><HR>
    Results:<br>
    <?php
        if ($result ==0):
            echo ("<B>Error ".odbc_error()."."odbc_errormsg()."</B>");
        elseif (odbc_num_rows($result)==0):
```

```

        echo("<B>Query ran successfully</B>");
    else:
    ?>
    <TABLE BORDER=1>
    <TR>
    <?php
        for ($i =0;$i <odbc_num_fields($result);$i++){
            echo("<TH>".odbc_field_name($result,$i+1). "</TH>");
        }
    ?>
    </TR>
    <?php
        while(odbc_fetch_into($result,$row_array)!=FALSE){
            echo("<TR>");
            for ($j =0;$j <odbc_num_fields($result);$j++){
                echo("<TD>".$row_array [$j] . " </TD>");
            }
            echo("</TR>");
        }
        echo("</TABLE>");
    endif;
    } elseif ($host || $database || $query) {
        echo("All three fields must be filled in for a query<BR>");
    } else {
        echo("Use PHP to run an SQL Query on an iSeries database:<BR>");
    }
    ?>
    <HR><BR>
    <FORM ACTION=" <?php echo($_SERVER['PHP_SELF']) ?>" METHOD=POST>
    <TABLE BORDER=1><TR>
    <TD>iSeries Host:</TD>
    <TD><INPUT TYPE=TEXT NAME="host" VALUE="<?php echo ($host);?>"></TD>
    </TR></TABLE>
    <BR>
    Library of the database to query:<BR>
    <INPUT TYPE=TEXT NAME="database" VALUE="<?php echo ($database);?>">
    <HR>
    Please enter the SQL query to be run:<BR>
    <TEXTAREA name="query" cols="40" rows="5">
    <?php echo ($query); ?>
    </TEXTAREA>
    <BR>
    <INPUT TYPE=SUBMIT VALUE="Execute query">
    </FORM>
    <p>View PHP <a href="dbqueryphp.php.source"target="_blank">Source for this Query</a>
    </BODY>
    </HTML>

```

The highlights include:

- ▶ **odbc_connect:** This is the “Open” of the database. The link variable is used by other ODBC functions later in the script.
- ▶ **odbc_exec:** The variable filled in on the HTML form contains the string that we run as an SQL statement. `odbc_exe` runs the SQL statement and returns results in the `$result` variable.
- ▶ **odbc_numfields:** A function determines the number of columns that are returned for this record. We use this value to place HTML `<TH></TH>` tags around each cell.

Another PHP script

For one additional PHP example, let us include a script that works only in the OS/400 PASE version of PHP. This example takes advantage of the fact that the OS/400 PASE “system” command writes any spooled output, produced by a command, to standard output. That is, you can run any commands with an OUTPUT(*PRINT) parameter in the OS/400 PASE shell and have the results sent to STDOUT.

For example, if you're on the OS/400 PASE command line QP2TERM, you can type the **system wrkactjob** command (for the Work with Active Jobs (WRKACTJOB) command of OS/400) and see the results as they scroll across the screen. Our example, `phpactjob`, simply formats this output into an HTML table. Figure A-5 shows the output of this script.

PHP Running WRKACTJOB in PASE

5722SS1 V5R1M0 010525 Work with Active Jobs 5/28/02										
Reset : *NO										
Subsystems : *ALL										
CPU Percent Limit : *NONE										
Response Time Limit : *NONE										
Sequence : *SBS										
Job name : *ALL										
CPU % . . . : .0 Elapsed time : 00:00:00 Active jobs										
-----Elapsed-----										
Subsystem/Job	User	Number	Type	Pool	Pty	CPU	Int	Rsp	AuxIO	Function
B19AUTH	QSYS	362108	SBS	2	0	.0			0	.0
AUTOSTART	B19AUTH	362109	ASJ	2	30	.0			0	.0 CMD-QSH
QPOZSPWT	B19AUTH	362112	BCI	2	30	.0			0	.0 PGM-QZSCHLI
QPOZSPWT	B19AUTH	362119	BCI	2	30+	28.2			0	.0 PGM-QZSCHLI
QZSHSH	B19AUTH	362110	BCI	2	30	.0			0	.0 PGM-QZSHSH
B10CHFD	QSYS	362133	SBS	2	0	.0			0	.0

Figure A-5 PHP formats the result of Work with Active Jobs (WRKACTJOB) as an HTML table

Example A-5 shows the `phpactjob` source code. Note that we use reverse single quotation marks (``) to run the Work with Active Jobs (WRKACTJOB) command and capture the output. This output is then broken into lines by searching for the new line character “\n” using the **strtok** function of PHP.

Example: A-5 Source code for PHPACTJOB

```
<html>
<head>
<title>PHPACTJOB Test PHP with WRKACTJOB Output</title>
</head>
<p>PHP Running WRKACTJOB in PASE<br>
<?PHP
$!sout=~/Q0penSys/usr/bin/system 'wrkactjob```;
```

```

$line = strtok($lsout, "\n");
print "<table border CELSPACING=0 CELLPADDING=0 BGCOLOR=\"#96FB84\">";
print "<tr>";
print "<td><pre>";
print $line;
print "</pre></td>";
print "</tr>";
while ($line = strtok("\n"))
{
    print "<tr>";
    print "<td><pre>$line</pre></td>";
    print "</tr>";
}
print "</table>";
print "thend";
?>
</body>
</html>

```

For more information

You have been introduced to PHP and how to use it on the iSeries server. Use this IBM Redbook as a starting point to find other examples and documentation that can have you running PHP in no time.

- ▶ PHP project Web site for help, tutorials, and examples about PHP
<http://www.php.net>
- ▶ OS/400 PASE on iSeries PartnerWorld Web site
<http://www-919.ibm.com/developer/factory/pase/overview.html>
- ▶ Linux on iSeries home page
<http://www.iseries.ibm.com/linux>
- ▶ A demo application showing PHP using ODBC Driver for Linux
<http://www-1.ibm.com/servers/eserver/iseries/linux/odbc/guide/demoindex.html>
 This demo includes PHP using binary large objects (BLOBs) that contain employee photos in the sample iSeries EMPLOYEE database.
- ▶ ASP to PHP Web site for ASP users who should consider migrating to PHP
<http://asp2php.naken.cc/>

Beware of PHP bugs

In the past, a few security holes were discovered in PHP. The most recently discovered one involves the code for handling file uploads. This flaw lets hackers easily crash the PHP server and possibly take it over remotely. The flaw affects PHP versions 4.2.0 and 4.2.1. CERT rates the problem as *critical*.

The PHP Group announced a fix release, version 4.2.2, that all PHP users employing PHP's file-upload facility should install immediately. The fix is available on the Web at:

http://www.php.net/release_4_2_2.php

Prerequisites

This section assumes that you have the following hardware and software on your iSeries server:

- ▶ **5722-SS1 OS/400 (5722-SS1) at V5R2:** The same basic steps should work on an iSeries server at V5R1
- ▶ **5722-SS1 Option 13 OS/400 System Openness Includes**
- ▶ **5722-SS1 Option 33 OS/400 PASE**

Note: In this IBM Redbook, we assume that you are running at V5R2 of OS/400. If you have OS/400 V5R2, then you must make sure that 5722-SS1 Option 33 OS/400 PASE is installed.

Since OS/400 V5R1 supports some levels of AS/400 hardware that are not supported by OS/400 PASE (requires a certain version (level) of PowerPC processor), you must first determine whether your AS/400 hardware supports OS/400 PASE. You can find a detailed list of processors on which OS/400 PASE can run on the Web at:

<http://www-919.ibm.com/servers/eserver/iseries/developer/factory/pase/ehardware.html>

- ▶ **5722-DG1 IBM HTTP Server for iSeries:** This Licensed Program Product (LPP) contains the HTTP Server (powered by Apache), which is the only HTTP server for which PHP works. Also, install the latest Apache group PTF package. For V5R2, the group PTF package number is SF99098.
- ▶ **The `make` command:** You can find the `make` command in OS/400 PASE for V5R2. If you are using V5R1 of OS/400, then you must download the `make` command. We recommend that you use the GNU `make` command that can download from:
<http://www.gnu.org/directory/gnu/make.html>
- ▶ **5799-PTL PRPQ iSeries Tools for Developers (Optional):** This toolkit is *optional* for this work, but you may find it useful for some other similar projects. For details, see:
<http://www.iseries.ibm.com/developer/factory/tools>

We also assume that you have the following hardware and software on your *build machine*. The build machine can be either a separate IBM @server pSeries® server running AIX or an iSeries running OS/400 with the following software:

- ▶ **The `patch` command:** The `patch` command is included in OS/400 PASE in V5R2. If you do not have a `patch` program on your system, try the GNU `patch`. The GNU `patch` program is usually not on AIX or OS/400 machines. You can download version 2.5 (not 2.5.4) from:
<ftp://ftp.gnu.org/pub/gnu/patch>
To compile the source, follow these steps:
 - Untar the source using the `tar` command.
 - Type `cd` to go to the directory.
 - Perform a `./configure`.
 - Run the `make` command.
 - Run the `make install` command.
- ▶ **GNU `gzip` command:** Compresses and decompresses files. You can download this from:
<http://www.gnu.org/directory/GNU/gzip.html>

- **VisualAge C++ compiler for AIX:** You can find information about this compiler at:

<http://www.ibm.com/software/ad/vacpp/>

If your build machine is AIX (not OS/400), you must match the AIX version to the target OS/400 PASE version. That is, the application binary created on AIX needs to be compatible with the version of OS/400 PASE in which you want the application to run. To help you plan this issue, see:

<http://publib.boulder.ibm.com/series/v5r2/ic2924/info/rzalf/rzalfplanning.htm>

We tested these instructions on AIX 4.3 and newer. Alternatively, V5R2 of OS/400 PASE now supports installation of either the IBM VisualAge C++ Professional for AIX Version 6.0 or the IBM C for AIX Version 6.0 software products. This means you can compile OS/400 PASE applications within OS/400 PASE. A separate AIX system is not required. IBM VisualAge C++ Professional for AIX Version 6.0 (5765-F56) and IBM C for AIX (5765-F57) are separately available program products from IBM. Note that the VisualAge C++ Professional for AIX compiler product also includes the C for AIX compiler product.

- **Perl scripting language:** This is needed to install VisualAge C++. You can download Perl to your iSeries from the Web at:

<http://www.cpan.org/ports/index.html#os400>

Installing PHP on the iSeries server

Follow these steps to download and prepare the PHP source files for compile.

Important: The installation steps described in the following sections use the PHP source package. This package is patched for DB2 UDB for iSeries and then compiled. However, we also found a Web page that provides PHP binary versions for iSeries. Go to:

<http://www.mcind.com/php/>

At the time of writing this publication, the Web site contained PHP up to version 4.3.5.

Pre-preparation for installation

If you want to perform the complete installation on your iSeries server, you have to install VisualAge C++ and Perl first.

Installing the Perl scripting language

Follow these steps:

1. Download the binary distribution (see “Prerequisites” on page 398 for the location of this code).
2. Place it under the /home/yourid directory. The file should have the name perl-5.8.0@18380-os400.tgz.
3. Start an OS/400 PASE terminal and change into the directory in which you placed the file.

Note: To get an OS/400 PASE (instead of a Qshell) terminal session, enter the following command from an OS/400 command line:

```
CALL QP2TERM
```

4. Use the following command to unzip the binary distribution:

```
gunzip perl-5.8.0@18380-os400.tgz
```

5. Change to the root directory (cd /) and run the command:

```
tar -xvf /home/yourid/perl-5.8.0@18380-os400.tar
```

The distribution is already archived from QOpenSys, so it places the files into the correct directory path.

6. To finish the Perl installation, generate symbolic links for Perl commands:

```
ln -s /QOpenSys/perl/bin/* /QOpenSys/usr/bin
```

Installing VisualAge C++

As soon as you finish downloading the trial version of VisualAge C++ and complete the Perl installation, install VisualAge C++ on your iSeries server:

1. Open a new, or return to your, OS/400 PASE terminal and create the source installation directory. We use vacpkg in this example:

```
mkdir /QOpenSys/vacpkg
```

2. Place the VisualAge C++ distribution into this directory (using File Transfer Protocol (FTP)). The file should be named vacpp.60.tnb.tar.Z.

3. In your terminal session, change to the directory /QOpenSys/vacpkg and extract the file:

```
gunzip vacpp.60.tnb.tar.Z
```

4. Use **tar** for extraction:

```
tar -xvf vacpp.60.tnb.tar
```

5. A work-around is needed to get a working installation and let the install script run correctly:

```
mv usr/sys/inst.images/vacpp.tnb usr/sys/inst.images/vacpp.lic  
mv usr/sys/inst.images/vac.tnb usr/sys/inst.images/vac.lic
```

6. Restore the script:

```
restore -qf usr/sys/inst.images/vacpp.ndi ./usr/vacpp/bin/vacppndi
```

7. Use Perl to install the application:

```
perl usr/vacpp/bin/vacppndi -d /QOpenSys/vacpkg/usr/sys/inst.images -b /QOpenSys/vac600
```

The script takes a while to complete. Finally it installs VisualAge C++ into the directory /QOpenSys/vac600.

8. Add the symbolic links for VisualAge:

```
ln -s /QOpenSys/vac600/usr/vacpp/bin/* /QOpenSys/usr/bin
```

After these two steps of pre-installation, you can continue with the installation of PHP itself.

Downloading PHP

Download the version of PHP you need for your iSeries server.

Note: We include the patch files for both the 4.3.0 and the older 4.2.2 versions of PHP. These instructions, however, are written for version 4.3.0.

1. Download the tar file php-4.3.0.tar.gz for PHP 4.3.0 from the following Web site:

<http://www.php.net>

2. Using FTP, send this file to the machine on which you will build PHP. This may be the AIX machine or the iSeries machine with the VisualAge compiler. We call this your *build machine*.

Tip: During our installation and testing, we noticed that the best location to place the php-4.3.0.tar.gz files is under the /home directory tree, when using your iSeries as the build machine. The following configuration installs PHP in the directory /QOpenSys/php.

3. Untar the file by using the following commands:

```
gunzip php-4.3.0.tar.gz
tar -xvf php-4.3.0.tar
```

Patching the source code file

A patch is required to run PHP on the iSeries server. We included patch files for both the 4.3.0 and the older 4.2.2 versions of PHP. The patch changes the default PHP DB2 support from AIX DB2 to OS/400 DB2.

1. Download and save the patch file to the build machine. See Appendix D, “Additional material” on page 421, for information about downloading the file. Download the patch file into the same directory from which you ran the **tar** command.
2. Change directory (**cd**) to that directory and run the following patch command:

```
cd php-4.3.0
patch -p1 < ../php430.patch
```

The -p1 says to remove a level from the patch filenames, so it looks for ext/odbc/php_odbc.c instead of php-4.3.0/ext/odbc/php_odbc.c.

This has the advantage that, if you download a version of PHP that has not changed too much from the one for which we provided a patch, it works. For example, you can actually use PHP 4.3.1 with the 4.3.0 patch because the files we patched did not change.

Locating iSeries-specific files

You must locate and bring to your build machine the following iSeries files:

- ▶ The sqlcli.h and the libdb400.exp files that contain DB2 UDB AS/400 support.
- ▶ The as400_libc.exp file is an iSeries-specific extension to the AIX file libc.a. This file is part of 5722-SS1 Option 13 OS/400 - System Openness Includes.

Follow these instructions to obtain these files from your iSeries server:

1. Enter the following command:

```
CPY OBJ('/QIBM/include/sqlcli.h') TODIR('/home/yourid') TOCCSID(*STDASCII) DTAFMT(*TEXT)
```

2. Using FTP, place the /home/yourid/sqlcli.h file from your iSeries server to the build machine.
3. Using FTP, send the *libdb400.exp* and *as400_libc.exp* files from the iSeries directory /QOpenSys/QIBM/ProdData/OS400/PASE/lib to the AIX build machine.

Note: Skip steps 2 and 3 if the build machine is your iSeries server.

Preparing for the PHP compile

Follow these steps to prepare the files and directories needed for the successful compile of PHP on your build machine. These steps assume that you are using **ksh**.

1. Set the CFLAGS, CC, and CPPFLAGS environment variables as follows. You *must* enter the **export CFLAGS='.....'** command all on one line. There is no “\” continuation character.:

Note: The flags for -l and -bl are the uppercase format of the letter “i”.

- If the build machine is an AIX server, enter:

```
export CFLAGS='-ma -DPASE -I /home/yourid -bI:/home/yourid/libdb400.exp
-bI:/home/yourid/as400_libc.exp'
export CC=xlc
export CPPFLAGS=-qflag=e:e
```

- If the build machine is your iSeries server, enter:

```
export CFLAGS='-ma -DPASE -I /home/yourid
-bI:/QOpenSys/QIBM/ProdData/OS400/PASE/lib/libdb400.exp
-bI:/QOpenSys/QIBM/ProdData/OS400/PASE/lib/as400_libc.exp'
export CC=xlc
export CPPFLAGS=-qflag=e:e
```

2. Change to the php-4.3.0 directory using the **cd** command.
3. Change the authority to Execute on the files config.guess and config.sub. You can do this by using the command:

```
chmod +x config.guess
chmod +x config.sub
```

4. Run the following command (it configures the script to install PHP in the directory /QOpenSys/php). The continuation character (‘\’) is not necessary if you type it all on one line.

```
./configure --with-ibm-db2 \
--with-config-file-path=/QOpenSys/php/etc \
--prefix=/QOpenSys/php/ \
--enable-force-cgi-redirect \
--without-mysql \
--disable-mysql
```

5. If you are compiling directly in OS/400 PASE on iSeries, add the following configure flags.

Note: The continuation character (‘\’) is not necessary if you type it all on one line.

```
--build=ibm-aix4.3.3.0 \
--host=powerpc-ibm
```

The configuration should take some time to run. After it finishes, you need to make final adjustments to the files listed in the following steps.

6. Edit the Makefile:

Note: The Makefile is generated with lines greater than 2048 characters. Some editors, such as vi, cannot handle the long lines, so you need to use a different editor. Send the Makefile, using FTP, to a different machine and back if necessary.

If you are compiling on your iSeries server, you can use the Edit File (EDTF) command, but be careful with lines, that go beyond the window size. If you change such lines, verify the correctness by using the Display File (DSPF) utility.

```
remove -ldb2 from ODBC_LIBS
remove -ldb2 from EXTRA_LIBS
```

7. Edit the config_vars.mk file:

Note: PHP version 4.3.0 does not have a config_vars.mk. This step is for PHP version 4.2.2 only.

```
remove -ldb2 from ODBC_LIBS
remove -ldb2 -lbind from EXTRA_LIBS
```

8. Edit the main/build-defs.h file:

Tip: You can see that the `./configure` command worked, when all of the configuration files that should be changed contain the mentioned statements.

```
remove -ldb2 from PHP_ODBC_LIBS
```

9. Edit the main/php_config.h file:

```
Delete #define HAVE_MMAP 1
Delete #define HAVE_SETITIMER 1
```

If you run this on a V5R1 OS/400 server, edit:

```
Delete #define HAVE_STATVFS 1
Delete #define HAVE_PREAD 1
Delete #define HAVE_PWRITE 1
```

Note: When editing text files (such as Makefile, php-config.ini, or any other script) in a Windows Notepad or WordPad, make sure to remove the carriage return (\r) from the file before you use it. You can do this in OS/400 PASE by using the commands:

```
tr -d "\r" < Makefile > Makefile.new
mv Makefile.new Makefile
```

Compile (make)

You have two choices depending on whether you are compiling in AIX on the pSeries server or in OS/400 PASE on the iSeries server.

Compiling in OS/400 PASE on the iSeries server

Note: To get a POS/400 ASE (instead of a Qshell) terminal session, enter the following command from an OS/400 command line:

```
CALL QP2TERM
```

Follow these steps if you are compiling the PHP source code on your iSeries server:

```
make
make install
mkdir /QOpenSys/php/etc
cp php.ini-dist /QOpenSys/php/etc/php.ini
```

This installs and puts all the files in the correct directory. You need write access to the /QOpenSys directory.

At this point, you may skip to “Testing PHP” on page 405.

Compiling in AIX on the pSeries server

Follow these steps if you are compiling the PHP source code on your pSeries:

1. Edit the Makefile (see the note in step 6 on page 403 about the long lines of the Makefile) for the line "install_targets =", remove "install-pear".

2. Enter the following commands in the order shown:

```
mkdir /tmp/QOpenSys
```

3. At the AIX prompt, run the following commands:

```
make
make install INSTALL_ROOT=/tmp/
```

This installs PHP into /tmp/QOpenSys/php.

4. Enter the following commands in the order shown:

```
mkdir /tmp/QOpenSys/php/etc
cp php.ini-dist /tmp/QOpenSys/php/etc/php.ini
```

5. Edit the Makefile (see the note in step 6 on page 403 about the long lines of the Makefile) for the line "install_targets =", add "install-pear".

If the location of your home directory on your AIX box is different than the location of your home directory in OS/400 PASE (for example, on AIX your home directory is /usr/home/usr4/jdoe and on OS/400 PASE it is /home/john), replace all occurrences of "/usr/home/usr4/jdoe/" to "/home/john/" in the Makefile. Make sure that you include the first and last "/" so you don't lose your directory separator.

6. Enter the following commands in the order shown:

```
cd /tmp
tar -cvf ~/php430pasebin.tar QOpenSys
cd ~
tar -cvf php430pasesrc.tar php-4.3.0
```

7. Using FTP, send both php430pasebin.tar and php430pasesrc.tar to your home directory on the iSeries server.

8. Enter the following commands in the order shown:

Note: The following steps are all done in OS/400 PASE on the iSeries and not in AIX.

```
cd /
tar -xvf ~/php430pasebin.tar
cd ~
tar -xvf php430pasesrc.tar
cd php-4.3.0
make install-pear
```

Testing PHP

From the OS/400 PASE shell, run the command:

```
/QOpenSys/php/bin/php -v
```

This should tell you the version of PHP you have.

Note: If you try running the PHP binary in OS/400 PASE and it dies with an illegal instruction, check for the existence of a job log. Several things can cause an illegal instruction signal and kill a OS/400 PASE application. If the illegal instruction was caused by an unsupported system call, the name of the system call is specified in the job log.

The job log should tell you the name of the illegal instruction. Next find the corresponding HAVE_ line in the php_config.h and delete it. Then recompile. This should only happen if you're compiling on a version of AIX that supports certain syscalls that OS/400 PASE does not support (in addition to the five noted earlier).

Configuring HTTP Server (powered by Apache) to use PHP

Edit the file httpd.conf using the Apache GUI interface. The key statements needed are:

```
ScriptAlias /php-bin/ /QOpenSys/php/bin/  
AddType application/x-httpd-php .php  
Action application/x-httpd-php /php-bin/php  
ServerUserID userprofile  
<Directory /QOpenSys/php/bin>  
    Options +ExecCGI  
    order allow,deny  
    allow from all  
</Directory>
```

Stop and start your HTTP Server (powered by Apache) Web server.

File permissions: If you can serve index.html without problems, but cannot serve index.php, this is most likely due to the fact that PHP is running as OS/400 user profile QTMHTTP1. QTMHTTP1 is the default OS/400 profile used for CGI applications. The default OS/400 profile for serving static files is QTMHHTTP, which most likely has the proper authorities.

To solve this problem, give access to both the file (read) and all the directories above the file (read and execute) in the IFS to the user profile QTMHTTP1.

Creating a sample database

Here we can add the creation of the sample database as supplied with the system. Starting in V5R1, a sample database is shipped with the system. This is explained on the DB2 Universal Database Web site at:

<http://www.ibm.com/servers/eserver/iseries/db2/sqldata.htm>

To unpack and create the sample database, invoke the procedure from any SQL interface:

```
CALL QSYS.CREATE_SQL_SAMPLE('SAMPLE')
```

Here SAMPLE is the name of the schema that you want to create. However, currently the sample PHP requires updates. For example, OS/400 PASE PHP runs as a CGI and cannot use the \$_SERVER ['PHP_AUTH_USER'] and \$_SERVER ['PHP_AUTH_PW'] values.

Also, when connecting to a database, you may normally use something like this example:

```
$dsn = "DRIVER=iSeries Access ODBC Driver;SYSTEM=$isdb_system;DBQ=$isdb_database";  
$db = odbc_connect($dsn, $user, $password);
```

In OS/400 PASE, you use something like this example:

```
$db = odbc_connect($isdb_system, "", "");  
odbc_setoption($db, 1, SQL_ATTR_DBC_DEFAULT_LIB, $isdb_database);
```

Note: It does not matter which user ID and password you use when you connect to the ODBC database. It uses the authority of the user profile that is running the Web server process. Use the ServerUserID directive in the Apache configuration to change this. It is actually somewhat of a security hole if you allow others to make a Web page and do not configure the Apache Web server to run the under a different user.

Limitations

Since PHP runs as a CGI application and not as an Apache module, some things do not work in this implementation on the iSeries server:

- ▶ HTTP authentication does not work, so any script that tries using the variables `$_SERVER['PHP_AUTH_USER']` and `$_SERVER['PHP_AUTH_PW']` does not work. You need to use user accounts and make a form that gets the user name and password and sets a cookie instead.
- ▶ `PHP_SELF` does not work. There is a bug in the CGI version of PHP 4.3.0 that corrupts the `$_SERVER['PHP_SELF']` variable. For more details about this bug, see the PHP page: <http://bugs.php.net/bug.php?id=21261>

By the time you read this, that page may have a patch that fixes the issue. If it does, then apply the patch. If it doesn't, use the fix suggested by *tapken@engter.de* in the bug report, which says to follow these steps:

- a. Create a file called "self_fix.php" in `/QOpenSys/php/lib/php/` with the following script:

```
<?  
    $_SERVER['SCRIPT_NAME'] = substr($_SERVER['PATH_TRANSLATED'],  
                                   strlen($_SERVER['DOCUMENT_ROOT']));  
    $PHP_SELF = $SCRIPT_NAME = $_SERVER['PHP_SELF'] = $_SERVER['SCRIPT_NAME'];  
?>
```

- b. In `/QOpenSys/php/etc/php.ini`, look for the line that says:

```
auto_prepend_file =
```

- c. Change this line to:

```
auto_prepend_file = self_fix.php
```

This should fix the `$_SERVER['PHP_SELF']` bug.

Note: Since this is a bug in PHP, there is no support if these steps do not solve the problem. See the restriction statement at the beginning of this chapter for more information.

- ▶ PHP and Net.Data both use the SQL Call Level Interface (CLI) application programming interface (API) to access the database. The problem arises when two different applications use the SQL CLI interface in the same job. Unfortunately, the SQL CLI provides no way to isolate different applications that use the SQL CLI. Within a job, there is only one SQL environment handle. Anyone who uses the SQL CLI uses the same environment handle.

When Net.Data performs a database operation and then PHP comes in to do the same, problems occur. PHP and Net.Data cannot coexist within the same CGI job if the both interact with the database. This is true for any CGI application that uses the SQL CLI.

If you want to use the same HTTP server for both applications, you can circumvent this by ensuring that Net.Data runs under one user profile and PHP under another. Unfortunately, about the only way to enforce this in your HTTP Server (powered by Apache) is to use basic authentication to force the HTTP server to adopt the authority of a different user profile for each of the clients. This, of course, is not always an option.

PHP 4.2.2 errata

The biggest change from 4.2.2 to 4.3.0 was the configuration process. To make this document apply to 4.2.2, make the following changes to the steps listed in the previously sections as noted:

- ▶ For “Locating iSeries-specific files” on page 401:
 - **Step 6 on page 403:** Ignore the note about the long line Makefile because it does not exist.
 - **Step 7 on page 403:** PHP version 4.3.0 does not have config_vars.mk. This step is for PHP version 4.2.2 only.
- ▶ For “Compiling in AIX on the pSeries server” on page 404, follow these steps instead. This is because it does not use PHP itself to try to install PEAR.

```
cd php-4.2.2
make
cd ..
tar -cvf php422pasesrc.tar php-4.2.2
```

Using FTP, send the tar file to OS/400 PASE.

The following steps are all done in OS/400 PASE on the iSeries and not in AIX:

```
cd ~
tar -xvf php422pasesrc.tar
cd php-4.2.2
make install
```


Bringing Tomcat Version 5.5 to your iSeries server

Tomcat is a servlet and JavaServer Pages (JSP) container that is used in the official Reference Implementation for the Java servlet and JSP technologies. The Java servlet and JSP specifications are developed by Sun under the Java Community Process.

Tomcat is developed in an open and participatory environment and released under the Apache Software License. Tomcat is intended to be a collaboration of the best-of-breed developers from around the world.

The iSeries supports Tomcat Version 3.2.4 as a component of 5722-DG1. See Table 2-2 on page 20 for details of the packaging on the iSeries. You can learn about the IBM supported version of the Tomcat server on the iSeries in 9.2, “Apache Software Foundation’s Jakarta Tomcat on iSeries” on page 197.

It may not be so surprising that the HTTP Server (powered by Apache) administrative GUI also uses the built-in Jakarta Tomcat servlet container engine at version 3.2.4 to generate the content.

Why would you spend the extra time and effort to bring Tomcat Version 5.5 to your iSeries server? Tomcat 5.5, compared to Version 3.2, offers a lot of enhancements for your iSeries server. It implements the Servlet 2.4 and JSP 2.0 specifications. The Eclipse Java Development Tools (JDT) is now the default compiler in Jasper. For a complete listing of the old and new functions, see the Jakarta Tomcat home page at:

<http://jakarta.apache.org/tomcat>

Jakarta Tomcat Version 5.5 on iSeries is not supported by IBM. We provide these instructions for you to download a public domain open-source copy of ASF Jakarta Tomcat so you can implement the new functions on your iSeries server. This chapter explains how to get the new version of Tomcat to your iSeries server.

Attention: Jakarta Tomcat Version 5.5 on iSeries is not supported by IBM. Use it at your own risk.

Software prerequisites

At the time of writing this redbook, the latest available version of Jakarta Tomcat was 5.5.1, which we use for this implementation. Tomcat 5.5.2 was still alpha code. Here is a detailed list of all software requirements:

- ▶ **5722-SS1 i5/OS at V5R3:** The same basic steps should work on an iSeries server at V5R2.
- ▶ **5722-SS1 Option 30 i5/OS - Qshell Interpreter**

Note: For releases prior to V5R3, an OS/400 PTF corrects several Qshell problems. We recommend that you install this PTF:

- ▶ V5R1: SI08114
- ▶ V5R2: SI08117

- ▶ **5722-DG1 IBM HTTP Server for iSeries:** This is not mandatory, because Jakarta Tomcat Version 5 has a built-in Web server that can serve the included examples. Nevertheless we highly recommend it, so that you can configure your HTTP Server (powered by Apache) to connect to the Tomcat servlet engine.
- ▶ **5722-JV1 Developer Kit for Java**
- ▶ **5722-JV1 Option 5 Java Developer Kit 1.3 or 5722-JV1 Option 5 Java Developer Kit 1.4:** The Jakarta Tomcat server needs it to run.
- ▶ **Binary distribution of Jakarta Tomcat version 5.5.1:** You can download it from:
<http://archive.apache.org/dist/jakarta/tomcat-5>
You may also want to check the following site for archived versions:
<http://archive.apache.org/dist/jakarta/tomcat-5/archive/>
- ▶ **Binary distribution of Jakarta Tomcat version 5.5.1 compatibility package:** You can find the compatibility package for a specific Tomcat version in the corresponding bin directory.

Tip: We used the binary ZIP distribution (file named jakarta-tomcat-5.5.1.zip and jakarta-tomcat-5.5.1-compat.zip) of Tomcat 5.5 for the installation. If you do not have ZIP available on your iSeries server, see Appendix C, “Bringing Zip and Unzip to OS/400 PASE and Qshell environments” on page 419.

- ▶ **Jakarta Tomcat connector mod_jk for AJP 1.3 on your iSeries:** You can download this (as a ready made save file) from:
<http://www.apache.de/dist/jakarta/tomcat-connectors/jk/binaries/iseries/>
Select the version of the zipped mod_jk file you need and download it to your PC. We tested with v1.2.6.

Installation

The installation procedure is divided into three steps as explained in the following sections:

1. Install Tomcat 5.5 on your iSeries server.
2. Install the Tomcat 5.5 compatibility package.
3. Start Tomcat 5.5 on the iSeries server.

4. Install mod_jk connector.
5. Configure your HTTP Server (powered by Apache).

Installing Tomcat 5.5 on your iSeries server

Follow these steps to install Tomcat 5.5 on your iSeries server:

1. Download the binary distribution of Jakarta Tomcat 5.5.1 to your iSeries and place it in the /home directory. Name this file jakarta-tomcat-5.5.1.zip.
2. Start a Qshell terminal by entering Start QSH (STRQSH) on the 5250 command line.
3. Enter the following command to make /home your current working directory:

```
cd /home
```

4. Enter the following command to unzip the file:

```
unzip jakarta-tomcat-5.5.1.zip
```

Unzip places the file in the /home directory as /jakarta-tomcat-5.5.1.

Tip: We used the binary ZIP distribution (file named jakarta-tomcat-5.5.1.zip) of Tomcat 5.5 for the installation. If you do not have ZIP available on your iSeries server, see Appendix C, “Bringing Zip and Unzip to OS/400 PASE and Qshell environments” on page 419.

Another method to unzip the jakarta-tomcat-5.5.1.zip file is to use **jar**. Assuming you put the zip file in a directory called /home, the following commands unzip it into /home using **jar**:

```
qsh
cd /home
jar -xf jakarta-tomcat-5.5.1.zip
```

5. Enter the following command to change into this directory:

```
cd /home/jakarta-tomcat-5.5.1/bin
```

6. Set the environment variables as shown in Figure B-1 using the Work with Object Links (WRKLNK) command. Start a new 5250 session to your iSeries. On the 5250 command line, enter:

```
WRKLNK '/home/jakarta-tomcat-5.5.1/bin/*'
```

Another option for this is to press F12 from the Qshell command line and then work with the 5250 command line. When you finish, enter STRQSH on the 5250 command line to restart your Qshell session. It returns you to where you left off earlier.

7. Edit the setclasspath.sh file. Select 2 (Edit). See Figure B-1:

- a. Add the following line:

```
export -s JAVA_HOME=/qibm/proddata/java400/jdk13
```

- b. Change the line to read:

```
if [ ! -r "$JAVA_HOME"/bin/java -o ! -r "$JAVA_HOME"/bin/javac ];
```

```

Browse : /home/jakarta-tomcat-5.5.1/bin/setclasspath.sh
Record :      1  of      49 by 14          Column :      1      79 by 79
Control :

....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
*****Beginning of data*****
# -----
# Set CLASSPATH and Java options
#
# $Id: setclasspath.sh,v 1.8 2004/07/26 22:01:19 markt Exp $
# -----
export -s JAVA_HOME=/qibm/proddata/java400/jdk13 7a
# Make sure prerequisite environment variables are set
if [ -z "$JAVA_HOME" ]; then
    echo "The JAVA_HOME environment variable is not defined"
    echo "This environment variable is needed to run this program"
    exit 1
fi
if [ ! -r "$JAVA_HOME"/bin/java -o ! -r "$JAVA_HOME"/bin/javac ]; then 7b
    echo "The JAVA_HOME environment variable is not defined correctly"

F3=Exit    F10=Display Hex    F12=Exit    F15=Services    F16=Repeat find
F19=Left   F20=Right

```

Figure B-1 Editing the `setclasspath.sh` file: Setting the environment variables

Note: We edited out the variable for the Java debugger because we did not have this option on the system. If you have this option on the system, this statement does not have to be edited. The statement we removed was:

```
! -r "$JAVA_HOME"/bin/jdb -o
```

- At this point, verify that ports 8080 and 8009 are available on your iSeries server. You can do this by using the 5250 command Work with TCP/IP Network Status (NETSTAT) and select option *CNN. If you need to change the port number on which the server listens, you can refer to the Jakarta Tomcat documentation at:

<http://jakarta.apache.org/tomcat>

Installing the Tomcat 5.5 compatibility package

Tomcat 5.5. requires, by default, the Java 2 Standard Edition Runtime Environment version 5.0 (also known as JDK 1.5) or later. To run Tomcat 5.5 on earlier versions of Java Runtime Environments (JRE), you need to install an additional package. This package is called the *compat package* and provides support to run Tomcat 5.5 under JDK 1.3 or JDK 1.4. Refer to "Software prerequisites" on page 410 for a description about how to obtain the package.

Follow these steps to install the Tomcat 5.5.1 compatibility package on your iSeries server:

- Download the binary distribution (jakarta-tomcat-5.5.1-compat.zip) of Jakarta Tomcat 5.5.1 compatibility package to your iSeries and place it in the /home directory. Name this file jakarta-tomcat-5.5.1-compat.zip.

Note: The directory to place the jakarta-tomcat-5.5.1-compat.zip file needs to be the same directory you placed the Tomcat 5.5.1 file (jakarta-tomcat-5.5.1.zip) in. The reason for this location is that when the compressed file will be unzipped, the unzip process assumes that the Tomcat 5.5.1 directory (jakarta-tomcat-5.5.1) already exists as a sub-directory of the store location. In this case, it is the /home directory.

2. Start a Qshell terminal by entering Start QSH (STRQSH) on the 5250 command line.
3. Make /home your current working directory by entering the command:

```
cd /home
```

4. Enter the following command to unzip the file:

```
unzip jakarta-tomcat-5.5.1-compat.zip
```

Unzip places the following files into the existing directory structure under /jakarta-tomcat-5.5.1:

- **jmx.jar**: Stored in /home/jakarta-tomcat-5.5.1/bin directory
- **xercesImpl.jar**: Stored in /home/jakarta-tomcat-5.5.1/common/endorsed directory
- **xml-apis.jar**: Stored in /home/jakarta-tomcat-5.5.1/common/endorsed directory

Starting Tomcat 5.5 on the iSeries server

You have prepared the iSeries server to start the Tomcat server. Perform the following steps to start Tomcat:

1. Return to the Qshell terminal.
2. Change the directory to the Tomcat bin directory.

```
cd /home/jakarta-tomcat-5.5.1/bin
```

3. Start the Tomcat server using the following command:

```
./startup.sh
```

Note: If you receive an error upon starting the server, check your authorities for these files. You can do this by entering the following command from the /bin directory:

```
ls -l
```

You must have read and execute authorities for all the files in this directory. If not, enter the following command to grant the proper authorities you need for these files:

```
chmod 755 *
```

You should see the startup environment variables being set as shown in Figure B-2.

```
$
> ./startup.sh
Using CATALINA_BASE:  /home/jakarta-tomcat-5.5.1
Using CATALINA_HOME:  /home/jakarta-tomcat-5.5.1
Using CATALINA_TMPDIR: /home/jakarta-tomcat-5.5.1/temp
Using JAVA_HOME:      /qibm/proddata/java400/jdk13
$
```

Figure B-2 Setting the startup environment

4. Verify your server is starting by entering the 5250 command Work with Active Jobs (WRKACTJOB). Look for the QP0ZSPWT job as highlighted in Figure B-3.

```

Work with Active Jobs                                     FRA821
                                                         10/04/04 14:41:53
CPU %:      .5      Elapsed time: 00:00:17      Active jobs: 222

Type options, press Enter.
  2=Change  3=Hold  4=End  5=Work with  6=Release  7=Display message
  8=Work with spooled files 13=Disconnect ...

Opt  Subsystem/Job  User      Type  CPU %  Function      Status
   QINTER          QSYS      SBS    .0      CMD-WRKACTJOB  DEQW
   QPADEV000F      BARLEN    INT    .1      CMD-WRKACTJOB  RUN
   QPADEV000G      BARLEN    INT    .0      CMD-TELNET     SELW
   QP0ZSPWT        BARLEN    BCI    .1      JVM-org.apache TIMW
   QZSHSH          BARLEN    BCI    .0      PGM-QZSHSH     TIMW
   QZSHSH          BARLEN    BCI    .0      PGM-QZSHSH     TIMW

                                                         Bottom

Parameters or command
===>
F3=Exit  F5=Refresh  F7=Find  F10=Restart statistics
F11=Display elapsed data  F12=Cancel  F23=More options  F24=More keys

```

Figure B-3 Job QP0ZSPWT verifies that your Tomcat 5.5 server is started

Your server should be operational after the status changes to TIMW.

5. To verify that your Tomcat server is operational and functioning, open a browser and enter the following Uniform Resource Locator (URL):

`http://your.server.name:8080`

You see the window shown in Figure B-4.

Note: Using the previous steps, the Tomcat server starts in the interactive subsystem. To start the Tomcat server in a batch mode, you can submit the job as follows:

```

SBMJOB CMD(QSH CMD('/home/jakarta-tomcat-5.5.1/bin/startup.sh')) JOB(TOMCAT551)
JOBQ(QSYSNOMAX)

```

You can end the Tomcat server in an orderly fashion by entering the following command on a 5250 command line:

```

SBMJOB CMD(QSH CMD('/home/jakarta-tomcat-5.5.1/bin/shutdown.sh')) JOB(TOMCATEND)
JOBQ(QSYSNOMAX)

```

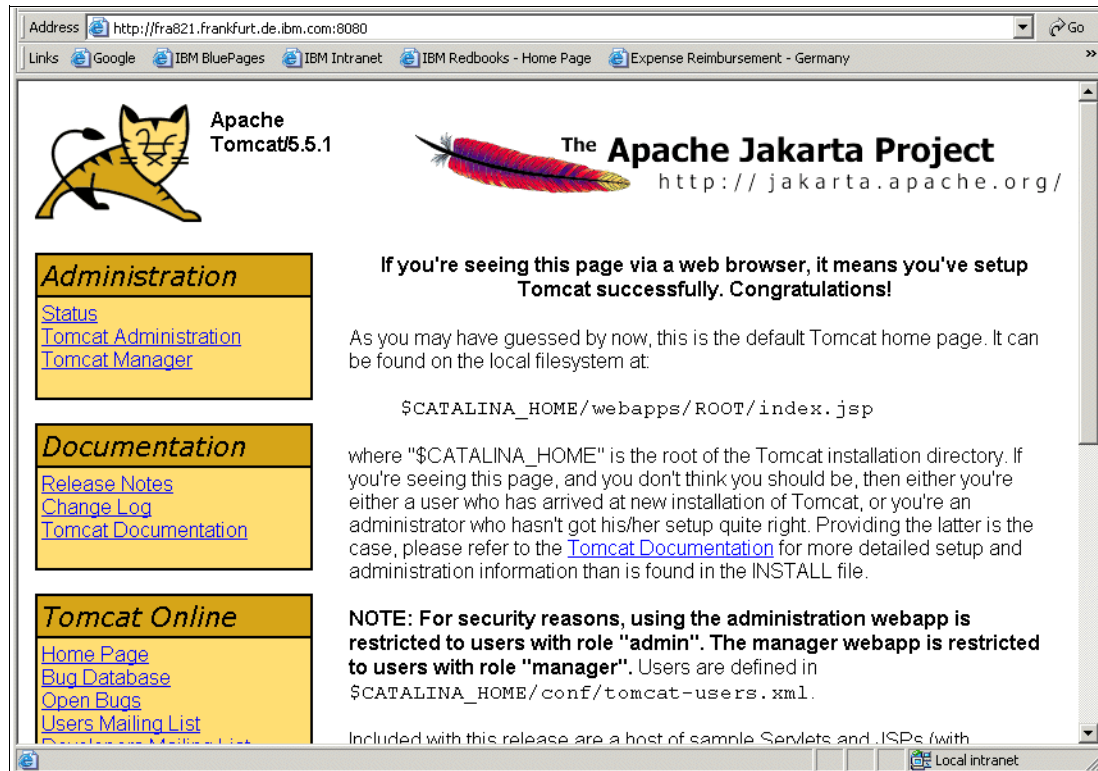


Figure B-4 Apache Jakarta Project Web page

You have now finished installing Tomcat on your iSeries server. The next step is to install the mod_jk connector.

Tip: After you work out the bugs from this process, you can use the Create Java Program (CRTJVAPGM) command and OPTIMIZE parameter to make the Jakarta TomCat 5.5 Java Archive (JAR) files run faster.

Installing mod_jk connector

If one application is written in Java using the servlet application programming interface (API), a connector is needed to route the requests from the Web server to the servlet engine. In this case, a Web server-specific connector is needed. We already have a connector on the iSeries, QZTCJK, which you can use with Tomcat V3.2.4. If you are using Tomcat V5.5, then this is an unsupported scenario.

Note: "Software prerequisites" on page 410 provides a link to download this connector. You can download this connector to your PC and then send it to your iSeries server using File Transfer Protocol (FTP). It is in the form of a save file (SAVF) and you have to restore it to library MOD_JK.

QZTCJK runs an older version of the AJP13 protocol. However older versions of the mod_jk plug-in, like the one on the iSeries, may run into problems with newer Tomcat versions because they enhanced the protocol. The only out-of-process Tomcat that QZTCJK is supported with is the one that is shipped with the iSeries. It is configurable by the administrative GUI. At this time, QZTCJK with the AJP13 protocol is not supported with Tomcat 5.5.

1. Unzip the MOD_JK_126.ZIP file on your PC.
2. Send the Save File (SAVF) to the iSeries using a Windows command line FTP as shown in Figure B-5. It is assumed that the SAVF was unzipped to the PC drive C:\.

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>ftp fra821
User (fra821.frankfurt.de.ibm.com:(none)): barlen
331 Enter password.
Password:
230 BARLEN logged on.
ftp> bin
200 Representation type is binary IMAGE.
ftp> quote site nam 1
250 Now using naming format "1".
ftp> put MOD_JK_126.SAVF /qsys.lib/qgpl.lib/mod_jk_126.savf
200 PORT subcommand request successful.
150 Sending file to member MOD_JK_126 in file MOD_JK_126 in library QGPL.
250 File transfer completed successfully.
ftp: 7687680 bytes sent in 15.04Seconds 4,71Kbytes/sec.
ftp>
```

Figure B-5 Sending the MOD_JK connector via FTP to your iSeries

3. Restore the SAVF to the iSeries. Enter the following 5250 command:

```
RSTLIB SAVLIB(MOD_JK) DEV(*SAVF) SAVF(MOD_JK_126)
```

Configuring your HTTP Server (powered by Apache)

The last step is to configure your HTTP server to use the new connector. To do this, you manually edit the configuration file of one of your existing servers. You need to add several lines to the configuration file:

```
LoadModule jk_module /QSYS.LIB/MOD_JK.LIB/MOD_JK.SRVPGM
JkWorkersFile /www/tomitsol/conf/workers.properties
JkLogFile /www/tomitsol/logs/jk.log
JkLogLevel debug
JKMount /jsp-examples/* worker1
```

Here are the steps to edit the configuration file as demonstrated in Figure B-6:

1. From the IBM Web Administration for iSeries interface, select the server instance.
2. In the left navigation frame, under Tools, select **Edit Configuration File** (not shown).
3. Add the LoadModule directive to the configuration file. This should be your first entry in the configuration file:

```
LoadModule jk_module /QSYS.LIB/MOD_JK.LIB/MOD_JK.SRVPGM
```


4. Add the following remaining four directives as shown in Figure B-6, immediately before the first directory directive:

```
JkWorkersFile /www/tomitsol/conf/workers.properties
JkLogFile /www/tomitsol/logs/jk.log
JkLogLevel debug
JKMount /jsp-examples/* worker1
```

Note: The JKLogLevel debug directive is not necessary for the server to work, but it is convenient to have if you experience any problems and need to debug your server.

Click **OK** to save your new configuration.

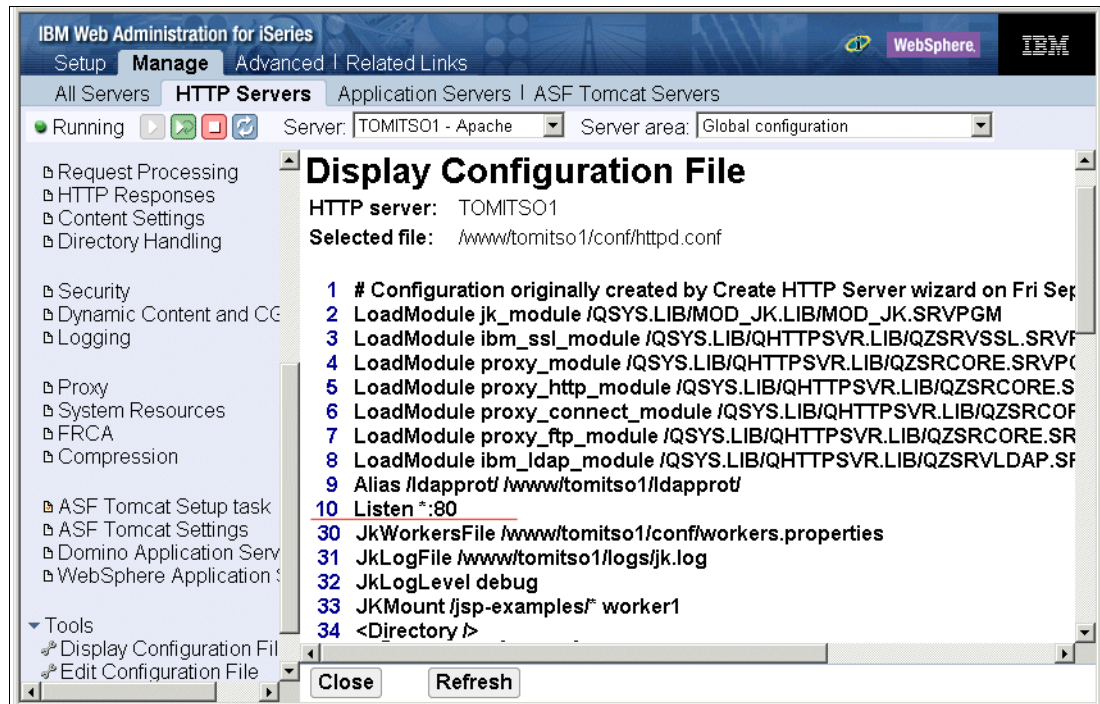


Figure B-6 Your new HTTP Server (powered by Apache) Tomcat out-of-process configuration

5. Copy the server.xml file from the Tomcat configuration to the HTTP Server (powered by Apache) configuration. This file is located in /home/jakarta-tomcat-5.5.1/conf. Copy it into into <ServerRoot>\conf.
6. Create a file named *workers.properties* in the same <ServerRoot>\conf directory in which you placed the server.xml file. Add the following lines of information:

```
worker.list=worker1
worker.worker1.type=ajp13
worker.worker1.port=8009
worker.worker1.host=fra821
```

7. Restart your HTTP Server (powered by Apache) and test your out-of-process Tomcat server. Use this URL in a Web client:

```
http://host.domain/jsp-examples/
```

You see the JSP Samples page as shown in Figure B-7.

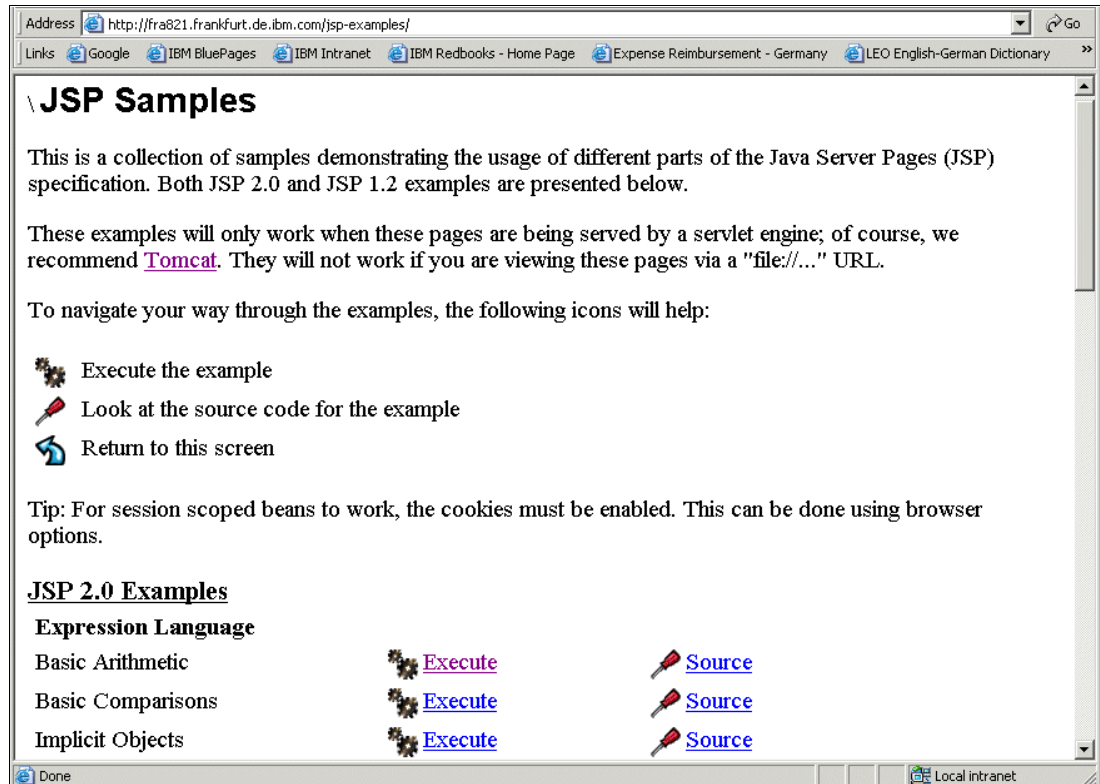


Figure B-7 Sample page from your HTTP Server (powered by Apache) and Tomcat server



Bringing Zip and Unzip to OS/400 PASE and Qshell environments

Ziping documents and directories has become common. It is also possible to use Zip functions on iSeries with a little help from AIX.

Important: IBM does not support Zip and Unzip on the iSeries server. We provide these instructions for you to download a public domain copy of the software tool so you can implement the new functions on your iSeries server.

Follow these steps to bring Unzip and Zip to your iSeries server:

1. Go to the following FTP site and download the archives:

<ftp://ftp.info-zip.org/pub/infozip/UNIX/AIX/>

In our case, we downloaded these files:

- **zip23x-aix43.zip**: For compressing (zip) files
- **unz550x-aix5L.tar.Z**: For uncompressing (unzip) files

2. Place these files somewhere on your iSeries server. We use /home/zip in this case.
3. Sign on to your iSeries and start an OS/400 Portable Application Solutions Environment (OS/400 PASE) terminal with the command:

```
CALL QP2TERM
```

4. Change into the directory, using the command:

```
cd /home/zip
```

5. The unzip file is a compressed file. Uncompress the file:

```
uncompress unz550x-aix5L.tar.Z
```

6. Untar the remaining file:

```
tar -xvf unz550x-aix5L.tar
```

It creates the directory unzip-5.50.

7. Change to this directory (**cd unzip-5.50**) and enter the **unzip** command. Now you can see whether it is working. To run the command from every directory, copy the unzip file into the directory /QOpenSys/usr/bin (PATH):

```
cp unzip /QOpenSys/usr/bin
```

8. Check whether unzip is really working. The Zip program on our system is packed, so we can try on that. Verify that you are in the directory in which the file zip23x-aix43.zip is located. Then enter the following command:

```
unzip -d ./zip zip23x-aix43.zip
```

This creates a directory named zip and places all files into that directory.

9. Change into this directory:

```
cd zip
```

10. Enter the **zip** command to see if the program is working.

11. For making the command available in every OS/400 PASE terminal session, copy the program into the /QOpenSys/usr/bin directory.

12. If you want to use both commands in Qshell, make another copy of the programs into the directory /usr/bin. The Zip and Unzip functions will be available for the next QSH terminal.



Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG246716>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select **Additional materials** and open the directory that corresponds with the redbook form number, SG246716.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
ReadMe-02.txt	Contains instructions on how to handle the files after you download them from the Internet.
tcp52dmast.zip	This is a zipped directory of /tcp52dmast and all its subdirectories. You can find all the ITSOco Web site and configuration files here.
tcp52lmast.savf	This is an iSeries Save File (*SAVF) object that contains library TCP52LMAST and other iSeries-specific objects to support some examples in this IBM Redbook. It was saved with a target release of V5R3.

tcp52lmast.savfv5r2	This is an iSeries Save File (*SAVF) object that contains library TCP52LMAST and other iSeries specific objects to support some examples in this IBM Redbook. It was saved with a target release of V5R2.
tcp52lmast.savfv5r1	This is an iSeries Save File (*SAVF) object that contains library TCP52LMAST and other iSeries specific objects to support some examples in this IBM Redbook. It was saved with a target release of V5R1.
php422pase.patch	This is a patch file used for Appendix A, "Bringing PHP to your iSeries server" on page 387.
php430pase.patch	This is a patch file used for Appendix A, "Bringing PHP to your iSeries server" on page 387.
php432pase.patch	This is a patch file used for Appendix A, "Bringing PHP to your iSeries server" on page 387.

The V4R5 and V5R1-based versions of this redbook also included examples. They are still available from the ITSO Web site.

<i>File name</i>	<i>Description</i>
ReadMe-00.txt	Contains instructions on how to handle the files after you download them from the Internet.
itsoDir.zip	This is a zipped directory of /ITSO and all its subdirectories. You can find all the ITSOco Web site and configuration files here.
itsoapache.savf	This is an iSeries Save File (*SAVF) object that contains library ITSOAPACHE and other iSeries-specific objects to support some examples in this IBM Redbook. It was saved with a target release of V5R1.
itsoapache.savfv4r5	This is an iSeries Save File (*SAVF) object that contains library ITSOAPACHE and other iSeries-specific objects to support some examples in this IBM Redbook. It was saved with a target release of V4R5.

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	6 MB minimum
Operating System:	Windows
Processor:	Any
Memory:	Any

How to use the Web material

Download the material and then follow the instructions found within the ReadMe-02.txt file.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 426.

- ▶ *IBM HTTP Server Powered by Apache on RS/6000*, SG24-5132
- ▶ *V4 TCP/IP for AS/400: More Cool Things Than Ever*, SG24-5190
- ▶ *Clustering and IASPs for Higher Availability on the IBM @server iSeries Server*, SG24-5194
- ▶ *Building AS/400 Internet-Based Applications with Java*, SG24-5337
- ▶ *Who Knew You Could Do That with RPG IV? A Sorcerer's Guide to System Access and More*, SG24-5402
- ▶ *AS/400 Internet Security: Implementing AS/400 Virtual Private Networks*, SG24-5404
- ▶ *Web Enabling AS/400 Applications with IBM WebSphere Studio*, SG24-5634
- ▶ *AS/400 HTTP Server Performance and Capacity Planning*, SG24-5645
- ▶ *AS/400 Internet Security: Developing a Digital Certificate Infrastructure*, SG24-5659
- ▶ *AS/400 Internet Security Scenarios: A Practical Approach*, SG24-5954
- ▶ *AS/400 XML in Action: PDML and PCML*, SG24-5959
- ▶ *Application Service Provider Business Model: Implementation on the iSeries Server*, SG24-6053
- ▶ *IBM @server iSeries Wired Network Security: OS/400 V5R1 DCM and Cryptographic Enhancements*, SG24-6168
- ▶ *Implementation and Practical Use of LDAP on the IBM @server iSeries Server*, SG24-6193
- ▶ *Domino and WebSphere Integration on the IBM @server iSeries Server*, SG24-6223
- ▶ *Managing OS/400 with Operations Navigator V5R1 Volume 1: Overview and More*, SG24-6226
- ▶ *Building Java Applications for the iSeries Server with VisualAge for Java 3.5*, SG24-6245
- ▶ *IBM @server iSeries Printing VI: Delivering the Output of e-business*, SG24-6250
- ▶ *Java and WebSphere Performance on IBM @server iSeries Server*, SG24-6256
- ▶ *Managing OS/400 with Operations Navigator V5R1 Volume 5: Performance Management*, SG24-6565
- ▶ *WebSphere Application Server V5 for iSeries: Installation, Configuration, and Administration*, SG24-6588
- ▶ *IBM Lotus Domino 6 for iSeries Implementation*, SG24-6592
- ▶ *IBM @server iSeries e-business Handbook: A V5R1 Technology and Product Reference*, SG24-6711

- ▶ *iSeries IP Networks: Dynamic!*, SG24-6718
- ▶ *WebSphere Development Studio Client for iSeries V5.0*, SG24-6961
- ▶ *OS/400 V5R1 Virtual Private Networks: Remote Access to the IBM @server iSeries Server with Windows 2000 VPN Clients*, REDP-0153
- ▶ *Enabling Web Services for the IBM @server iSeries Server*, REDP-0192
- ▶ *WebSphere Development Tools for iSeries Generating Web Front Ends to Existing Applications*, REDP-0516
- ▶ *WebSphere Application Server - Express V5.0 for iSeries*, REDP-3624
- ▶ *WebSphere for the IBM @server iSeries Server Server Buying and Selling Guide*, REDP-3646

Other resources

These publications are also relevant as further information sources:

- ▶ *HTTP Server for iSeries Programming*, GC41-5435
- ▶ *iSeries Performance Capabilities Reference Version 5, Release 3*, SC41-0607
<http://www-1.ibm.com/servers/eserver/iseries/perfmgmt/resource.htm>
- ▶ *Software Installation V5R2*, SC41-5120
- ▶ *Performance Tools for iSeries*, SC41-5340
- ▶ *TCP/IP Configuration and Reference*, SC41-5420
- ▶ Ahmad, Afrasiab; Antony, Mathew; Chittenden, Sean; Chopra, Vivek; Link, Michael; Sarang, Poornachandra; Wadlow, Stephen G.; Wainwright, Peter. *Professional Apache 2.0*. Wrox Press Inc., May 2002. ISBN 1-861007-22-1.
- ▶ Bloom, Ryan B. *Apache Server 2.0: The Complete Reference*. Osborne/McGraw-Hill, June 2002. ISBN 0-07-222344-8.
- ▶ Bowen, Rich et. al. *Apache Server: Unleashed*. Sams, 2000. ISBN 0-672-31808-3.
- ▶ Ford, Andrew and Estabrook, Gigi. *Apache: Pocket Reference*. O'Reilly & Associates, 2000. ISBN 1-56592-706-0.
- ▶ Kabir, Mohammed J. *Apache Server 2 Bible with CD-ROM*. John Wiley & Sons, March 2002. ISBN 0-7645-4821-2.
- ▶ Laurie, Ben et. al. *Apache: The Definitive Guide*. O'Reilly & Associates, 1999. ISBN 1-56592-528-9.

Referenced Web sites

These Web sites are also relevant as further information sources:

- ▶ IBM HTTP Server for iSeries
<http://www.ibm.com/eserver/iseries/software/http>
- ▶ IBM HTTP Server for iSeries documentation
<http://www-1.ibm.com/servers/eserver/iseries/software/http/docs/doc.htm>
- ▶ TCP/IP for OS/400
<http://www.ibm.com/servers/eserver/iseries/tcpip>

- ▶ *Software Installation Guide*
<http://www.ibm.com/series/infocenter>
- ▶ Apache Software Foundation
<http://www.apache.org>
- ▶ *ApacheToday* news and information online
<http://www.apachetoday.com>
- ▶ *Apache Week* news and information online
<http://www.apacheweek.com>
- ▶ Onlamp news and information online
<http://www.onlamp.com/apache/>
- ▶ Web-based Distributed Authoring and Versioning (WebDAV)
<http://www.webdav.org/>
- ▶ iSeries Network
<http://www.iseriesnetwork.com>
- ▶ IGNITE/400 iSeries On Demand Business user group
<http://www.ignite400.com>
- ▶ Net.Data manuals and documentation
<http://www.ibm.com/eserver/series/software/netdata/docs/doc.htm>
- ▶ Sample CGI programs
<http://www.ibm.com/eserver/series/software/http/examples/>
- ▶ Easy400, CGI Web development tools Web site for iSeries
<http://www-922.ibm.com/easy400p/easy400p01.html>
This site includes a link to download the CGIDEV2 ILE-RPG CGI Development Toolkit.
- ▶ i/net makes a series of Web servers for the iSeries server
<http://www.inetmi.com/series/>
- ▶ Netcraft
<http://www.netcraft.com/survey/>
- ▶ NetObjects
<http://www.netobjects.com/>
- ▶ The IBM @server and IBM TotalStorage Lab Services
<http://www.ibm.com/servers/eserver/services/>
- ▶ IBM developerWorks
<http://www.ibm.com/developerworks>
- ▶ IBM alphaWorks
<http://www.alphaworks.ibm.com/>
- ▶ IBM PartnerWorld
<http://www.ibm.com/partnerworld>
- ▶ Many excellent third-party Web sites focus on different aspects of the iSeries. Here is a partial list:
 - Search400.com
<http://search400.techtarget.com/>

- IGNITe/400
<http://www.ignite400.org/>
- Common
<http://www.common.org/>
- Midrange Computing Press Online
<http://www.mcpressonline.com/>
- iSeries Network
<http://www.iseriesnetwork.com/>
- @server Magazine, iSeries edition (formerly *iSeries Magazine*)
<http://eservercomputing.com/iseries/>
- ▶ SPECweb99 from Standard Performance Evaluation Corporation (SPEC)
 - <http://www.specbench.org/osg/web99/>
 - <http://www.specbench.org/osg/web99ssl/>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols

*VLDL 54
+MultiViews 384
.htaccess 62, 102

Numerics

2058 e-business Cryptographic Accelerator 300
3DES 139
404 "Page Not Found" 305
4758 e-business Cryptographic Coprocessor 300
 national language support (NLS) 381

A

access control 101, 102
access log reporting 10
AccessFileName (directive) 62
active threads 303
Add Cluster Node Entry (ADDCLUNODE) command 365
Add TCP/IP Interface (ADDTCPIFC) command 362
AddLanguage (directive) 384
AddOutputFilterByType (directive) 250
Administrative Console 39, 49
administrative GUI
 delete 27
 enabling SSL 137
 enhancements 33
 Manage 38
 manage 27
 national language support (NLS) 376
 rename 27
 restart 27
 Setup 37
 start 27
 stop 27
 TCM 56
Advanced Single Server Edition 193
AES 138
AIX binaries 388
ajp12 (Tomcat) 200
ajp13 (Tomcat) 200
alias (directive) 80, 91
All Servers tab 37
Allow (directive) 102
AllowCONNECT (directive) 143
AllowOverride (directive) 62, 102, 104
Apache history 3
Apache Portable Runtime (APR) 12, 311
 see modules
Apache server version 351
Apache, market share 4
APACHEDFT 41, 43
Application Servers tab 37
application, ITSOCO 24

APR (Apache Portable Runtime) 12, 311
ASF Jakarta Tomcat, see Tomcat
ASF Tomcat Servers 37, 40
asynchronous I/O 14, 228
authentication 6, 55, 101
 by a validation list 108
 by Kerberos tickets 120
 by LDAP entries 113
 by OS/400 user profiles 105
 failure 325
authentication failure 326
AuthName (directive) 104
authorities for administration 22
authorities for Tomcat 201
AuthType (directive) 104
automatic expiration management 11
Autostart 43

B

Base Edition 193
Base64 105
Base64 encoding 108
basic authentication 6, 103
 LDAP 113
 OS/400 user profile 105
 validation list 108
benchmark, SPECweb99 223
binaries, AIX 388
BrowserMatch 244
buckets and brigades 313
bytes received 305
bytes sent 305

C

cache processing (seconds) 305
cache responses 305
cache target, TCM 263, 272
cached (seconds) 305
CacheLocalFD (directive) 237
CacheLocalFile (directive) 237
CacheLocalFileMmap (directive) 237
CacheLocalSizeLimit (directive) 236
CERN 3, 173
CGI 302
 initialization at server startup 234
Change CRG Primary (CHGCRGPRI) command 369
Change HTTP Attributes (CHGHTTTPA) command 52
Change Job Queue Entry (CHGJOBQE) command 364
Change Network Attributes (CHGNETA) command 362
Change TCP/IP Attributes (CHGTCPA) command 232
CHGCRGPRI command 369
CHGHTTTPA command 52
CHGJOBQE command 364
CHGNETA command 362

- CHGTCPA (Change TCP/IP Attributes) command 232
- child job, multi-threaded 228
- cipher 138
- cipher suite list 138
- client authentication 7
- client side digital certificate 139
- clustered hash table 356
- clustering, see high availability (HA)
- collection services 12, 346
- Command Module Structure 318
- Common Gateway Interface (CGI) 9, 160
 - Net.Data 161
 - PHP 388
- Common Tasks and Wizards 37, 50, 178
- communications trace 256, 353
- compat package 412
- compression 13, 240, 241
 - BrowserMatch 244
 - by MIME type 250
 - DEFLATE 242
 - input filters 241
 - logging 252
 - output filters 241, 242
 - SetEnvIf 244
- conf/httpd.conf 46
- CONFIG 178
- configuration directives 59
- configuration directory listings 63
- configuration recommendations 63
- configuration structure 60
- container areas 42
- contexts 59, 60
 - directory 60, 64
 - file 60
 - global context 60
 - location 60
 - VirtualHost 60, 75, 76, 80, 95, 128
- cookies 339
- co-processors, cryptographic 300
- copy into memory method 237
- Create Cluster (CRTCLU) command 364
- CRTCLU command 364
- Cryptographic Accelerator, 2058 300
- Cryptographic Access Provider 19
- Cryptographic Coprocessor, 4758 300
- customer module 302

D

- data compression 241
- data source, TCM 263, 272
- DB2WWW 161
- DEFAULT 178
- defending the IFS, see security
- Delete Communications Trace (DLTCMNTRC) command 291, 353
- delete HTTP server 27
- denial of service, see performance
- Deny (directive) 102
- DES 138
- digital certificate 53

- Digital Certificate Manager (DCM) 19, 127
 - national language support (NLS) 375
- directive
 - AccessFileName 62
 - AddLanguage 384
 - AddOutputFilterByType 250
 - alias 80, 91
 - Allow 102
 - AllowCONNECT 143
 - AllowOverride 62, 102, 104
 - AuthName 104
 - AuthType 104
 - CacheLocalFD 237
 - CacheLocalFile 237
 - CacheLocalFileMmap 237
 - CacheLocalSizeLimit 236
 - configuration 59
 - Deny 102
 - DocumentRoot 76, 80, 91
 - DynamicCache 238
 - ErrorLog 76, 80, 91
 - FRCACacheLocalFileStartUp 290
 - FRCACookieAware 297
 - FRCAEnableFileCache 290
 - FRCAEnableProxy 294
 - FRCAEndofURLMarker 297
 - FRCAMaxCommBufferSize 298
 - FRCAMaxCommTime 298
 - FRCAProxyCacheRefreshInterval 294
 - FRCAProxyPass 292, 294
 - FRCARandomizeResponse 298
 - HAModel 367
 - HostNameLookups 231
 - HotBackup 229
 - JkAsfTomcat 199
 - JkLogFile 199
 - JkLogLevel 199
 - JkMount 199
 - JkMountCopy 199
 - JkWorkersFile 199
 - KeepAliveTimeout 231
 - LanguagePriority 384
 - LimitRequestBody 233
 - LimitRequestFields 233
 - LimitRequestFieldSize 233
 - LimitRequestLine 233
 - LimitXMLRequestBody 233
 - Listen 76, 80, 91, 128
 - LiveLocalCache 237
 - LmURLCheck 367
 - LoadModule 128, 143, 197
 - LogMaintHour 333
 - mod_status 345
 - NameVirtualHost 89, 91, 95
 - Order 62, 102
 - PasswdFile 104
 - ProxyNoConnect 143
 - ProxyPass 143
 - ProxyPassReverse 143
 - ProxyReceiveBufferSize 143

- ProxyRequests 143
- ProxyVia 143
- Require 102
- ServerAdmin 76, 80, 91
- ServerAlias 92
- ServerName 76, 80, 91
- SetEnvIfNoCase 244
- SSLAppName 128
- SSLCacheDisable 128
- SSLCipherSpec 138
- SSLEnable 128
- SSLVersion 138
- ThreadsPerChild 228
- Tomcat 199
- UseCanonicalName 96
- UserID 104
- VirtualDocumentRoot 95, 97
- VirtualDocumentRootIP 95
- VirtualScriptAlias 95
- VirtualScriptAliasIP 95
- Directive Index 48
- directory context 60
- directory listing configuration 15, 63
- directory name interpolation 95
- directory walk 61
- Display Cluster Information (DSPCLUINF) command 364, 365
- Display CRG Information (DSPCRGINF) command 369, 370
- Display Network Attributes (DSPNETA) command 361
- Display Subsystem Description (DSPSBSD) command 236, 363
- Display System Value (DSPSYSVAL) command 376
- Display the Configuration File Tool 46
- Display User Profile (DSPUSRPRF) command 375
- DLTCMNTRC command 291, 353
- DMPCMNTRC command 353
- DMPUSRTRC command 341
- document base directory 199
- document list 56
- DocumentRoot (directive) 76, 80, 91
- Domino plug-in 11
- DSPCLUINF command 364, 365
- DSPCRGINF command 369
- DSPNETA command 361
- DSPSBSD command 236, 363
- DSPSYSVAL command 376
- DSPUSRPRF command 375
- Dump Communications Trace (DMPCMNTRC) command 353
- Dump User Trace (DMPUSRTRC) command 341
- dynamic content 235
- dynamic data 157
 - allow SSI to call CGI 160
 - Common Gateway Interface (CGI) 160
 - Net.Data 161
 - server-side includes (SSI) 158
- dynamic virtual hosting 6
- DynamicCache (directive) 238

E

- Edit File (EDTF) command 47
- Edit Object Authority (EDTOBJAUT) command 108
- EDTF command 47
- EDTOBJAUT command 108
- EIM (Enterprise Identity Mapping) 114, 121
- encryption key 138
- encryption protocol 138
- encryption, see security
- End Communications Trace (ENDCMNTRC) command 291, 353
- ENDCMNTRC command 291, 353
- Enterprise Identity Mapping (EIM) 114, 121
- error responses 305
- ErrorLog (directive) 76, 80, 91
- European Laboratory for Particle Physics (CERN) 173
- expiration management, automatic 11
- External Cache Communication Protocol (ECCP) 264

F

- Fast Response Cache Accelerator (FRCA) 281
 - directives, all 296
 - implementation
 - local cache 288
 - reverse proxy 292
 - introduction 282
 - limitations 286
 - local cache hit scenario 284
 - local cache miss scenario 283
 - Network File Cache (NFC) 287
 - reverse proxy hit scenario 286
 - reverse proxy miss scenario 285
- file context 60
- forward proxy 142, 143
- FRCA Proxy 303
- FRCA Stats 302
- FRCA, see Fast Response Cache Accelerator (FRCA)
- FRCACacheLocalFileStartUp (directive) 290
- FRCACookieAware (directive) 297
- FRCAEnableFileCache (directive) 290
- FRCAEnableProxy (directive) 294
- FRCAEndofURLMarker (directive) 297
- FRCAMaxCommBufferSize (directive) 298
- FRCAMaxCommTime (directive) 298
- FRCAProxyCacheRefreshInterval (directive) 294
- FRCAProxyPass (directive) 292, 294
- FRCARandomizeResponse (directive) 298

G

- General Server Configuration 43
- global context 60
- Global Server Settings 51
- GO LICPGM command 374
- group 54
- group file 52, 54
- group PTFs 21
- GUI configuration and administration, create HTTP Server 24
- GUI, Admin, see administrative GUI

H

- HAModel (directive) 367
- handshaking, SSL 137
- hash algorithm 138
- headers control 15
- high availability (HA) 14, 355
 - clusters 356
 - implementation 359
 - liveness monitor 356
 - packaging 19, 20
 - peer model 359
 - primary or backup with a network dispatcher model 358
 - primary or backup with takeover IP model 356, 359
 - takeover IP 356
- HostNameLookups (directive) 231
- HotBackup (directive) 229
- HTTP Server (original) 6, 20
- HTTP Server (powered by Apache) 20
 - testing 24
- HTTP Server overview
 - access log reporting 10
 - Apache Portable Runtime (APR) 12
 - CGI 9
 - delete 27
 - Domino plug-in 11
 - dynamic virtual hosting 6
 - features 4
 - high availability 14
 - LDAP 9
 - local memory cache 8
 - manage 27
 - mod_deflate 13
 - persistent connection 5
 - proxy caching 7
 - PTFs 24
 - rename 27
 - requirements 18
 - restart 27
 - reverse proxy caching 8
 - server-side includes (SSI) 9
 - software installation 23
 - start 27
 - stop 27
 - Tomcat 11
 - TRCTCPAPP 12
 - Triggered Cache Manager (TCM) 13
 - Version 1.1 5
 - virtual host 6
 - Web usage mining 10
 - Web-based Distributed Authoring and Versioning (WebDAV) 10
 - Webserver Search Engine 10
 - WebSphere Application Server plug-in 11
- HTTP Server statistics 39
- HTTP server trace 257, 341
- HTTP Servers tab 37
- HTTP Version 1.1 5
- HTTP virtual host 72
- httpd.conf 46

hypertext pre-processor, see PHP

I

- I/O, asynchronous 14, 228
- i5/OS 174
- IASP 14
- IBM HTTP Server for iSeries 18
- IBM i5/OS 174
- IBM Tivoli Web Administration tool 114
- IBM Tivoli Web Site Analyzer 340
- idle threads 303
- IFS security, see security
- IIS, Microsoft 4
- in-context configuration 60
- index 56
- Information Center 36
- Information Development xv
- inline protection 189
- inline protection setup 189
- in-process, see Tomcat
- Internet Daemon (INETD) 363
- Internet Printing Protocol (IPP) for NLS 380
- Internet Server Provider (ISP) 96
- Internet Users and Groups 51, 52
- IP takeover 356
- IP-based
 - implementation of virtual hosts 77
 - virtual host 74
- iSeries Navigator 5, 24, 362
- iSeries Network xv, 388
- iSeries Tasks page 34, 35
- iSeries Tasks page for NLS 375
- ITSOco Web application 24

J

- J2EE 191
- Jakarta Tomcat, see Tomcat
- Java 2 Enterprise Edition (J2EE) 191
- Java Developer Kit (JDK) 18, 20
- jk_module 197
- JkAsfTomcat (directive) 199
- JkLogFile (directive) 199
- JkLogLevel (directive) 199
- JkMount (directive) 199
- JkMountCopy (directive) 199
- JkWorkersFile (directive) 199
- jni (Tomcat) 200

K

- Keep the file descriptor open method 237
- KeepAliveTimeout (directive) 231
- Kerberos 120
 - keytab 123
 - service accounts 124
- key length 138

L

language support, see national language support (NLS)

- language, for Web administration GUI 35
- LanguagePriority (directive) 384
- LDAP 9
 - authentication 113
 - authentication error 325, 326
 - IBM Tivoli Directory Server Web Administration tool 114
 - store HTTP configurations 9
- level of Apache server 351
- Licensed Program Product (LPP), installing 23
- LimitRequestBody (directive) 233
- LimitRequestFields (directive) 233
- LimitRequestFieldSize (directive) 233
- LimitRequestLine (directive) 233
- LimitXMLRequestBody (directive) 233
- Listen (directive) 76, 80, 91, 128
- LiveLocalCache (directive) 237
- liveness monitor 356
- LmURLCheck (directive) 367
- LoadModule (directive) 128, 143, 197
- local cache, see performance
- location context 60
- log expiration 333
- log maintenance 333
- LogCycle 11
- logging 11, 230, 331
- LPP (Licensed Program Product) 23

M

- Manage Application Server 50
- manage HTTP server 27
- Manage page GUI, see administrative GUI
- Manage tab 37
- mass dynamic 74, 94
- MD5 138
- memory map of the file method 237
- Microsoft IIS 4
- migration (original to Apache)
 - directives and services not supported 175
 - equivalent directives 176
 - functional differences 176
 - how to 177
 - new directives 176
 - report details 183
 - testing 188
- MIME (Multipurpose Internet Mail Extensions) 241, 313, 381
 - mod_deflate 13, 240
 - mod_header.c 315
 - mod_jk 415
 - mod_status 345
- modules 312
 - compile, link, export 319
 - debugging 322
 - design overview 312
 - example source code 315
 - how to 315
- Monitor Server option 39
- multi-homed server 72
- Multipurpose Internet Mail Extensions (MIME) 241, 313, 381
- multi-threaded child job 15, 228

N

- name-based
 - implementation 89
 - virtual host 74
- named protection 189
- NameVirtualHost (directive) 89, 91, 95
- National Center for Supercomputing Application (NCSA) 3, 173
- national language support (NLS) 373
 - 4758 Cryptographic Coprocessor 381
 - administrative GUI 376
 - Digital Certificate Manager (DCM) 375
 - Internet Printing Protocol (IPP) 380
 - iSeries Tasks page 375
 - secondary languages, install 374
 - Web site 381
- NCSA (National Center for Supercomputing Application) 3, 173
- ND server 39, 50
- Net.Data 20, 161
 - logs 340
 - macro 56
- Netcraft survey 4
- NetObjects' Fusion 24
- Network Authentication Service (NAS) 123
- Network Deployment (ND) server 39, 50
- network dispatcher 358
- Network File Cache (NFC) 287
- non-cache processing (seconds) 305
- non-cache responses 305
- non-cached (seconds) 305
- normal connections 303

O

- object dependency graph (ODG), TCM 263
- On Demand Business 191
- on demand business 191
- Operations Navigator 5
- Options +MultiViews 384
- Order (directive) 62, 102
- origin server 285
- OS/400
 - installing options 23
 - integration 3
 - software products and options 18
 - user profile authentication 105
- OS/400 Portable Application Solutions Environment (OS/400 PASE) 19, 387, 388
- out-of-process, see Tomcat

P

- Page Not Found 305
- PasswdFile (directive) 104
- pattern 60
- peer model 359

- performance 223
 - asynchronous I/O 228
 - CGI initialization at server startup 234
 - collection services 12, 346
 - components of 226
 - denial of service 233
 - dynamic caching 237
 - Fast Response Cache Accelerator (FRCA) 281
 - global parameters 227
 - HostNameLookups 231
 - HotBackup 229
 - KeepAliveTimeout 231
 - local cache
 - how to 238
 - introduction to 8
 - tracing 239
 - logging 12, 230
 - mod_deflate 13, 240
 - TCP buffer size 232
 - threads 228
 - Triggered Cache Manager (TCM) 259
 - zen of 226
- performance tools reports 348
- Perl 157
- persistent connection 5
- phases 313
- PHP 399
 - binary version 399
 - bugs 397
 - code example 390, 393, 396
 - configure HTTP Server 405
 - installation on iSeries 399
 - limitations 406
 - on iSeries 391
 - prerequisites 398
 - what is 388
 - why 389
- powered by Apache, see Apache
- primary or backup with a network dispatcher model 358
- primary or backup with takeover IP model 356, 359
- Print Communications Trace (PRTCMNTRC) command 291, 353
- problem determination 323
 - HTTP server trace 341
 - logging 331
 - Net.Data 340
 - startup parameters 351
 - status codes 352
- profile swapping 142
- property form 42, 43
- Proxy 302
- proxy caching 7
- proxy chaining 154
- proxy server, see security
- ProxyNoConnect (directive) 143
- ProxyPass (directive) 143
- ProxyPassReverse (directive) 143
- ProxyReceiveBufferSize (directive) 143
- ProxyRequests (directive) 143
- ProxyVia (directive) 143

- PRTCMNTRC command 291, 353
- PTFs 21, 24

Q

- Qshell Interpreter 20, 419
- question mark icon 36
- QZHAPREG (register application with DCM) 131, 137

R

- RC4 138
- Real Time Server Statistics 13, 49, 175, 301
 - Absolute tab 303
 - Averages tab 305
 - Delta 303
 - General tab 303
- realm 103
- recommendations, configuration 63
- Redbooks Web site 426
 - Contact us xv
- register application with DCM (QZHAPREG) 131, 137
- rename HTTP server 27
- request handling 313
- request routing 59, 61
 - .htaccess 62
 - directory walk 61
 - example 62
- requests 303, 304
- requests rejected 303
- Require (directive) 102
- requirements for HTTP Server 18
- response handler 313
- responses 303, 304
- restart HTTP server 27
- reverse proxy 142, 145, 285, 292
- reverse proxy caching 8

S

- scripting
 - CGI 160
 - language 387
 - Net.Data 161
 - Perl 157
 - PHP
- search engine 307
- search index 56
- Search Setup 51
- secondary languages 374
- security 101
 - access control 101, 102
 - administrative GUI SSL 137
 - authentication 101
 - Base64 105
 - basic authentication 103
 - LDAP 113
 - OS/400 user profile 105
 - validation list 108
 - client side digital certificate 139
 - encryption 7, 102

- forward proxy 142, 143
- proxy chaining 154
- proxy server 142
- reverse proxy 142, 145
- SSL client authentication 139
- SSL handshaking 137
- TLS upgrade 136
- server area 42, 48
- server authentication 7
- server handled 302
- server performance, see performance
- Server Properties 43
- server root directory 46
- server trace 257, 341
- ServerAdmin (directive) 76, 80, 91
- ServerAlias (directive) 92
- ServerName (directive) 76, 80, 91
- server-side 387
- server-side includes (SSI) 9, 158
- SetEnvIfNoCase (directive) 244
- Settings 51
- Setup page GUI, see administrative GUI
- Setup page, GUI
 - see administrative GUI
- SHA 138
- Shared Memory Control (QSHRMEMCTL) 23
- software installation for HTTP Server 23
- SPECweb99 benchmark 223
- SSI to call CGI 160
- SSL 302
- SSL client authentication 139
- SSL connections 303
- SSL encryption, see security
- SSL_RSA_WITH_3DES_EDE_CBC_SHA 139
- SSL_RSA_WITH_DES_CBC_SHA 139
- SSLAppName (directive) 128
- SSLCacheDisable (directive) 128
- SSLCipherSpec (directive) 138
- SSEnable (directive) 128
- SSLVersion (directive) 138
- Start Communications Trace (STRCMNTRC) command 291, 353
- start HTTP server 27
- Start PDM (STRPDM) command 163
- Start Performance Tools (STRPFRT) command 348
- Start TCP/IP Server (STRTCPSVR) command 341
- startup parameters 351
- static content 235
- status codes, HTTP 352
- stop HTTP server 27
- STRCMNTRC command 291, 353
- STRPDM command 163
- STRPFRT command 348
- STRTCPSVR command 341
- SunONE 4
- Survey, Netcraft 4

T

- takeover IP 356
- task 42

- Tasks, iSeries 34
- Tasks, iSeries for NLS 375
- TCM page, GUI
 - see administrative GUI
- TCM tab 51
- TCM, see Triggered Cache Manager (TCM)
- TCP send buffer size (TCPSNDBUF) 232
- TCP/IP configuration 72
- TCP/IP Connectivity Utilities 18
- testing 24
- thread safe 15
- threads 228
- ThreadsPerChild (directive) 228
- TLS encryption, see security
- Tomcat 11, 40, 302
 - ajp12 200
 - ajp13 200
 - authorities 201
 - comparison with WebSphere Application Server 194, 195
 - directives 199
 - directory structure 198
 - how it works 197
 - in-process 197, 198, 203
 - jk_module 197
 - jni 200
 - log files 202
 - out-of-process 197, 198, 208
 - overview 194
 - packaging 20
 - version 5.5 409
 - workers.properties 200, 220
- Toolbox for Java 20
- tools 42
- total (seconds) 305
- trace communications 256
- Trace TCP/IP Application (TRCTCPAPP) command 12, 341
- trace, HTTP server 257, 341
- trigger handler, TCM 262, 273
- Triggered Cache Manager (TCM) 13, 19, 51, 259
 - authorization 261
 - cache target 263, 272
 - data source 263, 272
 - directory structure 261
 - documentation 261
 - hosts 271
 - how it works 262
 - implementation 264
 - object dependency graph (ODG) 263
 - packaging 20
 - system requirements 260
 - trigger handler 262, 273
 - trigger message 265

U

- Uniform Resource Locator (URL) 34
- Unzip 419
- URL 34
- UseCanonicalName (directive) 96

- user authentication 101
- user profile 52
- user profile authorities 22
- UserID (directive) 104

V

- V5R3 174
- validation list 52, 54
- validation list authentication 108
- verify TCP/IP Connection (PING) 364
- version of Apache server 351
- virtual host 6, 71, 80
 - how to
 - IP-based 78
 - mass dynamic 95
 - name based 90
 - IP-based 74
 - IP-based implementation 77
 - mass dynamic 74, 94
 - name based 74
 - name-based implementation 89
 - overview 75
 - SSL/TLS encryption 128
- Virtual IP Address (VIPA) 72, 362
- VirtualDocumentRoot (directive) 95, 97
- VirtualDocumentRootIP (directive) 95
- VirtualHost context 60
- VirtualScriptAlias (directive) 95
- VirtualScriptAliasIP (directive) 95
- VisualAge C++ 400
- VT100 client 275

W

- WAR (Web Application Archive) 197
- Web Administration for iSeries 5, 33, 35
- Web application 171, 191
 - ITSOco 24
- Web Application Archive (WAR) 197
- Web application serving 191
- Web browser 22
- Web Cache Control Protocol (WCCP) 264
- Web Crawler 307, 309
- Web server market share 4
- Web server performance, see performance
- Web usage mining 10
- Web-based Distributed Authoring and Versioning (Web-DAV) 10
- WebDAV 10
- Webserver Search Engine 10, 20, 55, 307
- Webserver Web Crawler 20
- WebSphere 302
- WebSphere Administrative Console 39, 49
- WebSphere Application Server 20
 - Advanced Edition 193, 195
 - Base Edition 193
 - comparison with Tomcat 194, 195
 - express 193
 - Network Deployment Edition 193
 - overview 193

- plug-in 11
 - Single Server Edition 195
 - Standard Edition 193
- WebSphere Development ToolSet 19
- WebSphere Edge Server 358
- wizard 42
- wizard to create the HTTP Server 24

- work area frame 57
- Work with Active Jobs (WRKACTJOB) command 294
- Work with Object Links (WRKLNK) command 47
- Work with Relational Database Directory Entries (WRKRDBDIRE) display 164
- Work with System Values (WRKSYSVAL) command 23
- workers.properties, Tomcat 200, 220
- WRKACTJOB command 294
- WRKLNK command 47
- WRKSYSVAL command 23

X

- X.500 directory 113

Z

- Zeus 4
- Zip 419
- zlib 240
- ZOC/Pro 275



IBM HTTP Server (powered by Apache): An Integrated Solution for IBM @server iSeries Servers

(0.5" spine)

0.475" <-> 0.873"

250 <-> 459 pages



IBM HTTP Server (powered by Apache)

An Integrated Solution for IBM *e*server iSeries Servers



Fully exploit the integrated power of IBM i5/OS and Apache

Study the new administration GUI, SSL proxy, security, and compression

Extend ASF Jakarta Tomcat 5.5, PHP, APR, and modules

This IBM Redbook helps you to plan, install, configure, troubleshoot, and understand the HTTP Server (powered by Apache) running on the IBM *e*server iSeries server. It introduces the HTTP Server (powered by Apache) and identifies all the necessary components to install and configure your first Apache-based Web server running on your iSeries server. It includes a quick guide to the Apache contexts and request routing. It also introduces the iSeries' unique graphical user interface (GUI) for further configuration and customization.

This redbook explains how to use virtual hosts, secure your server, and serve dynamic data with server-side includes (SSI), Common Gateway Interface (CGI), Net.Data, and Hypertext Preprocessor (PHP). It details the steps required to implement Web application serving with Java featuring the Apache Software Foundation's (ASF) Jakarta Tomcat. Advanced topics include how to achieve the best performance by using local caches, mod_deflate, Triggered Cache Manager (TCM), and Fast Response Cache Accelerator (FRCA).

This redbook also introduces the Webserver Search Engine, problem determination, high availability (HA), and national language support (NLS) considerations. It includes an example of extending the core features of your HTTP Server (powered by Apache) via Apache Portable Runtime (APR) support.

To complete the discussion, this redbook includes appendices about bringing PHP and Tomcat Version 4.1 to your iSeries server, and bringing Zip and Unzip functions to the OS/400 Portable Application Solutions Environment (OS/400 PASE) and Qshell environments.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**