

ibm.com



e-business

# A V5R2 Look at the HTTP Server (powered by Apache)

EP01

ITSO iSeries Technical Forum

Brian R. Smith



# Redbooks

International Technical Support Organization

© 2003 IBM Corporation



# Acknowledgments



This presentation is based upon original work from the HTTP team at the IBM Rochester development lab. Their contributions are gratefully acknowledged. It was organized and expanded during a residency at the ITSO Rochester Center by Sadamitsu Hayakawa of IBM Japan and Brian R. Smith, ITSO Rochester.

Copyright International Business Machines Corporation 2000

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following terms are trademarks or registered trademarks of the IBM Corporation in the United States or other countries or

ADSTAR	BrioQuery	Information Warehouse	NetView	SmoothStart
Advanced Function Printing	BRMS	Integrated Language Environment	OfficeVision	SystemView
AFP	Client Series	Intelligent Printer Data Stream	OS/2	
AIX	DataGuide	IPDS	OS/400	
AnyNet	DataPropagator	iSeries 400	PowerPC	
Application Development	DB2	JustMail	PowerPC AS	
APPN	IBM (or e(logo)server)	Net.Commerce	Print Services Facility	
AS/400	IBM	Net.Data	PSF	
AT	IBM Network Station	NetFinity	SanFrancisco	

cc:Mail, Lotus, Lotus Notes, Lotus Domino, Domino.Action, and Domino.Merchant are trademarks or registered trademarks of Lotus Development Corporation in the United States or other countries or both.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

IBM's VisualAge products and services are not associated with or sponsored by Visual Edge Software, Ltd.

Intel and Pentium are trademarks of Intel Corporation in the United States and other countries.

Tivoli is a registered trademark of Tivoli Systems Inc. in the United States or other countries or both.

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

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific statement of direction.

Any performance data contained in this document was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurement quoted in this presentation may have been made on development level systems. There is no guarantee these measurements will be the same on generally available systems. Some measurements quoted in this presentation may have been estimated through extrapolation. Actual results may vary. Users of this presentation should verify the applicable data for their specific environment. Customer examples cited are examples of how the referenced customers use IBM and other products. Results vary by environment and may not be realized in all situations.

THIS MATERIAL IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. IN NO EVENT WILL IBM BE LIABLE TO ANY PARTY FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES FOR ANY USE OF THIS MATERIAL INCLUDING, WITHOUT LIMITATION, ANY LOST PROFITS, BUSINESS INTERRUPTION, LOSS OF PROGRAMS OR OTHER DATA ON YOUR INFORMATION HANDLING SYSTEM OR OTHERWISE, EVEN IF WE ARE EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Functions and availability dates may change after this presentation was completed.

© 2003 IBM Corporation



# **An Introduction to the HTTP Server (powered by Apache)**

# Notes



This part introduces the HTTP Server (powered by Apache) beginning with its history and includes information that leads you to understand its features and advantages.

- ▶ History of Apache
- ▶ Apache Benefits to iSeries Users
- ▶ Product Availability and Packaging
- ▶ Server Features

# History of Apache



- Most HTTP servers originated from CERN or NCSA
- Two different implementations provide similar services
- Prior to 2001 IBM's HTTP server products were CERN-based
- Apache is NCSA-based
- Name originated from "a patchy NCSA server"
  - Ongoing development by network of volunteers
  - Apache version 1.3 is very popular; version 2.0.43 is available
- Apache Software Foundation formed in 1995
- Apache is available on 30+ platforms
- Platforms that provide all source code to users are true "Apache"
  - IBM does not provide source code
  - IBM HTTP products are "powered by Apache"

# Notes



Most 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 had 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. In 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 the 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 API features. The latest level available for the Apache server is Version 2.0.43 which was recently GAed in April of 2002.

The market share for top servers across all domains in February 2002 was:

Source: February 2002 Netcraft (<http://www.netcraft.com/survey/>)

- 58.43% Apache
- 29.13% Microsoft IIS
- 2.92% iPlanet (Netscape)

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 and is available at on the Web at: <http://www.apache.org>

# Apache Benefits to iSeries Users



**Open source**

**Feature packed server**

**IBM's strategic server**

**Porting third party modules**

**Large installation base**

**Apache is a (relatively) secure server with tremendous Rochester developer support**

- **Problem:** CERT Advisory CA-2002-17 Apache Web Server Chunk Handling Vulnerability: Original release date: June 17, 2002
- **Solution:** As of Friday afternoon, June 28, the iSeries Apache development team has approved the following two hyper PTFs as a response to the subject security advisory. Elapsed days: 11

# Notes



**Open source** - iSeries Powered by Apache server is based on the open-source server code provided by the Apache Software Foundation. This version is based on the 'alpha' code for Apache version 2.0 - it will be updated as future Apache versions are made available. While iSeries source code will not be published, IBM will offer 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 will provide defect support for the iSeries Powered by Apache server. IBMers have long been active in Apache development.

**Feature packed server** - HTTP/1.1 compliant server

**IBM's strategic server** - In line with IBM Strategy on all eServers, and open Web Serving.

**Porting third party modules** - Apache is extremely customizable. IBM provides support for porting third-party and user Apache modules to the iSeries. The API for modules has changed significantly for 2.0. Many of the module-ordering/-priority problems from 1.3 should be gone. 2.0 does much of this automatically, and module ordering is now done per-hook to allow more flexibility. Also, new calls have been added that provide additional module capabilities without patching the core Apache server. The net of this is that first people who have modules written for 1.3 of Apache will have to port to version 2.0. But then, the port to the iSeries (as a service program) should be fairly straight forward.

**Large installation base** - Apache is already a well established server in the Web Server world. Recent Netcraft Web Survey says that Apache has around 60% of Market share. This will help finding already experienced Web Masters on this product. This also helps in the area of image - it is easier to justify putting a Web server on your iSeries. And, a large install base will allow Apache to have a high degree of quality. A wide user base for testing and fixing brings you a higher quality product.



# Notes



**Problem:** CERT Advisory CA-2002-17 Apache Web Server Chunk Handling Vulnerability: Original release date: June 17, 2002  
Last revised: --  
Source: CERT/CC

## Systems Affected:

- Web servers based on Apache code versions 1.3 through 1.3.24
- Web servers based on Apache code versions 2.0 through 2.0.36

## Overview

There is a remotely exploitable vulnerability in the handling of large chunks of data in web servers that are based on Apache source code. This vulnerability is present by default in configurations of Apache web servers versions 1.3 through 1.3.24 and versions 2.0 through 2.0.36. The impact of this vulnerability is dependent upon the software version and the hardware platform the server is running on.

## I. Description

Apache is a popular web server that includes support for chunk-encoded data according to the HTTP 1.1 standard as described in RFC2616. There is a vulnerability in the handling of certain chunk-encoded HTTP requests that may allow remote attackers to execute arbitrary code.

The Apache Software Foundation has published an advisory describing the details of this vulnerability. This advisory is available on their web site at

[http://httpd.apache.org/info/security\\_bulletin\\_20020617.txt](http://httpd.apache.org/info/security_bulletin_20020617.txt)

# Notes



**Solution:** (elapsed days: 11)

As of Friday afternoon, June 28, the iSeries Apache development team has approved the following two hyper PTFs as a response to the subject security advisory. The fix pertains to the 2.0.18 level of Apache that is currently shipped in OS/400 v4r5 and v5r1. The team is also working on porting 2.0.39 to v5r2 (Apache Software Foundation fixed the problem in 2.0.39). The fix will be incorporated into v5r2 when 2.0.39 is regression tested and officially delivered in PTF form - most likely the week of July 15. The PTF build team is working on transmitting the PTFs to retain this afternoon.

The APAR number for the problem is SE06465:

- OS/400 V4R5: SF67411
- OS/400 V5R1: SI04996

As of Monday morning July 1, users can also get an entire package of the most current HTTP server PTFs by ordering the appropriate group PTF. The aforementioned Apache integrity PTFs will be included in the refresh of the group PTFs. To be certain you have the fix, the data area for the group PTFs should contain a date of June 27, 2002:


- OS/400 V4R5: SF99035
- OS/400 V5R1: SF99156

External inquiries can be directed to the HTTP server web site for more information. Updates are scheduled to occur there by Monday, July 1:  
<http://www.ibm.com/eserver/series/software/http>

**Note:** In addition, the advisory reported other problems with the Apache server including the possibility that hackers could introduce their own code to the system. It should be noted that unlike some other platforms, iSeries was only vulnerable to a Denial of Service attack - it is not possible for hackers to introduce their own code via this exposure. This is due to the iSeries unique architecture.

# New V5R2 Enhancements



- Fast Response Cache Accelerator (FRCA)
- Highly Available HTTP server
- Log rollover and archival
- Logging to QSYS files
- TLS upgrade
- GUI enhancements (usability/accessibility/wizards) 
- Support for PASE CGI
- Full function migration wizard
- Supported by Domino 6
- Improved HTTP server management capabilities
  - Server statistics to collection services
- Inbound admission rate control with QoS



# Notes



Details on all of these topics are spread throughout this presentation.

**Inbound admission rate control with QoS** is not covered in this presentation. At a high level the new QoS control of inbound rate control is as follows:

Basic Connection Admission Control mechanism

- Limit acceptance rate of new connections
- Prioritized listen queue
- Keyed off local/remote IP address/ports

Basic URL Admission Control mechanism

- Limit acceptance rate of URL pages
- Prioritized queue
- Keyed off HTTP data plus local IP address/port
- Requires FRCA (Fast Response Cache Accelerator)
  - Available with V5R2 HTTP Server (powered by Apache) only

# Product Availability and Packaging



## V4R5

- Group PTF SF99035 to enhance 5769-DG1 product required to enable Apache. Now both:
  - HTTP Server for iSeries (original) and
  - HTTP Server for iSeries (powered by Apache)

## V5R1 and V5R2

- Built into base of 5722-DG1. Contains both:
  - HTTP Server for iSeries (original) and
  - HTTP Server for iSeries (powered by Apache)

## At V4R5, V5R1 and V5R2:

- Packaged as part of the same product
- Can coexist on the same iSeries server
- Based upon Apache 2.0. GA version for V5R2; PTFed back to V5R1
- Migration from original to HTTP Server (powered by Apache) available

# Notes



The V4R5 version of Apache was first shipped in December, 2000 via Group PTF SF99035 and was based on 'alpha' code from the Apache Software Foundation. Through a series of revisions and updates from both the source code (from the Apache Software Foundation) and the HTTP Server (powered by Apache) we have seen the progress from alpha; through beta; and now GA that coincides with V5R2.

V4R5 will stay at the 2.0.18 version of the Apache server - which is basically the beta version of Apache. If you want the GA version of the HTTP Server (powered by Apache) you will need to upgrade to V5R1 at the very least.

V5R2 will see GA version (currently 2.0.43) - the latest available from ASF. You can obtain the latest and greatest code by ordering the HTTP Server PTF Group SF99098. The PTF group is updated periodically with the latest PTFs. The PTF group number does not change. During an update, any additional PTFs that impact the HTTP server are added to this PTF group.

V5R2 function will be PTFed back to V5R1, but not V4R5. You can obtain this list of PTFs by ordering the HTTP Server Group PTF SF99156. The group PTF is updated periodically with the latest PTFs. The group PTF number does not change. During an update, any additional PTFs that impact the HTTP server are added to this group PTF. This group PTF will provide Apache 2.0.43 with all the new V5R2 function (except FRCA and HTTP Server collection services) to V5R1.

# Server Features: General functions



- Both** HTTP version 1.1 support
  - Persistent connections
- pbA** Support for the TRCTCPAPP command
- pbA** Support for iASPs

# Notes



**HTTP Version 1.1 support-** Both products support HTTP Version 1.1. The HTTP protocol implementation in Apache was chiefly architected by one of the HTTP version 1.1 authors. Most current versions of popular Web browsers support HTTP Version 1.1. Apache is normally configured to detect popular browsers that do not properly support HTTP Version 1.1, and use only HTTP Version 1.0.

- **Persistent connections-** When you enter a 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.

**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.

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

**Support for iASPs-** An independent auxiliary storage pool (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. So, while the system is active, you can take the IASP offline, bring it online, or switch between systems. IASPs contain any of the following:

- 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). It is not supported on the HTTP Server (original).



# Server Features: Dynamic Contents



**Both** Server-side includes (SSI)

**Both** CGI programming

 Control number of CGI jobs started with server and their user profile

 OS/400 PASE CGI programs (UNIX binaries)

 **User written modules**

 **Apache Portable Runtime (APR)**

 **Original server API**

# Notes



**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.

**CGI programming-** The start up how the CGIs behave is different with Apache as compared to the original server.

With Apache, a new child job is spawned each time a CGI is executed until the 'MaxCGIJobs' number is reached. MaxCGIJobs defaults to the value for 'ThreadsPerChild' which is set to 80 in your configuration (the default). Once 'MaxCGIJobs' is reached, the server looks for an available CGI child job to use. If none are available, it recycles the first inactive CGI job that it finds. The CGI jobs never end (until the server ends). Since the inactive CGI jobs consume virtually no system resources, it is OK to have them there. It is also possible to pre-start a large number of CGI jobs to eliminate the overhead of starting new ones, using the StartCGIJobs directive.

So, it is very likely that Apache users will see up to 80 CGI jobs (server BCI jobs) running at any given time. Idle jobs do not consume any significant resources, and using them in this manner reduces the startup cost for new CGI jobs.

At V5R1, we are also enforcing user profiles in CGI jobs. A CGI job can only have one user profile associated with it now. This will also result in a greater number of CGI jobs.

The iSeries can run a CGI application that has been written and compiled for AIX. The binary output of the compiler is moved to the iSeries and can run from the Portable Application Solution Environment (PASE). This built into the support for V5R2. For V5R1 you must load the latest group PTF for 5722-DG1. You can obtain this list of PTFs by ordering the HTTP Server Group PTF SF99156. The group PTF is updated periodically with the latest PTFs.

**User written modules** - The design of the HTTP Server (powered by Apache) is one that 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:

- Window's Dynamic Link Libraries (DLL)
- UNIX's shared object libraries
- OS/400's ILE Service Programs

# Notes



(continued)

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

You can extend the core functionality of the HTTP Server (powered by Apache) by writing your own modules. Or, take advantage of the many open source solutions written for the Apache server. Here is an example of a tool that can give you a 90% improvement for your iSeries Websphere application using a HTTP server plugging called MOD-GZIP that is now available from the eCTC

More: the mod-gzip compression software is a plugging today available from the internet for Intel platforms. The eCTC has done work on this freeware and tested it on the iSeries: we have prepared a documentation for you to have the mod-gzip on your iSeries HTTP server up within half a day. This enables at very little cost:

improved performance: customer sat / machine load increased

consolidation of the HTTP server inside the iSeries (in another LPAR): machine load increased

Full Details: Use the attached documents word or pdf+txt + html (configuration information) to have modgzip up immediately: the files explain how to download the code, modify and compile the mod\_gzip module to run on OS/400 V5R1.

For more eCTC support, please contact us !

> Eric Aquaronne > eServer Custom Technology Center

> e-Business Solutions Center (eBSC), IBM Europe/Middle-East/Africa Advanced Technical Support, France

> Tel. = Office: +33-4-9211-5791, Mobile: +33-686-086-401, email= aquaronn@fr.ibm.com, Fax: +33-4-9324-7821

> eCTC email: CTCEMEA@FR.IBM.COM

> eCTC Internet site: <http://www.ibm.com/servers/eserver/series/service/ctc/>

> eCTC Intranet site: <http://w3.rchland.ibm.com/projects/CustomTechnologyCenter>

note: the 90% figure quoted above is real, it has been provided by an EMEA ISV using modgzip on iSeries (replacing an NT server)

# Notes



**Apache Portable Runtime (APR)** - APR provides a set of routines to write your own modules to extend the functionality of the HTTP Server (powered by Apache). When you write your own module to add some function to your HTTP server instance, you can use APR and once you write it, it can be compiled on other platforms to run.

To serve dynamic contents can be one of the solutions achieved by using APR.

**Original Server API- HTTP Server (original)** APIs are not supported on HTTP Server (powered by Apache). The strategic direction of IBM is to extend the function of the Web server using Java servlets rather than with modules or server APIs. This function can only be used in HTTP Server (original).

# Server Features: Performance & Cache



**Both** Asynchronous I/O

**Both** KeepAliveTimeout (works with persistent connections)

**Both** Denial of service

**Both** Local memory cache

**pbA** Fast Response Cache Accelerator (FRCA)

**Both** Proxy caching

**pbA** Reverse Proxy cache

**Both** Triggered Cache Manager (TCM)

# Notes



**Asynchronous I/O-** With the HTTP Server (powered by Apache and original) implementation, the HTTP Server processes communications requests asynchronously. In this asynchronous I/O model, threads are only involved in processing when there is work to be done. Threads are dispatched to perform work as required and 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 utilizing 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.

**KeepAliveTimeout-** When the server runs with persistent connections, the KeepAliveTimeout setting determines the number of seconds the server waits for subsequent requests before closing the connection. If this value is too low, the server could be impacted in terms of performance as connections could be closed frequently. If this value is too high, the Web server could have many connections open and the server could run out of resources. In this case the use of asynchronous I/O can alleviate (but not eliminate) the problem of running out of resources.

**Denial of service-** The denial of service attribute is equally a performance setting as well as a security setting. This setting allow us 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 to the one it expects. Although this setting impacts the server performance as each request is tracked, it allow 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.

**Local memory cache-** 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 is significantly faster than if the file is retrieved from disk storage.

**Fast Response Cache Accelerator (FRCA)-** With the HTTP Server (powered by Apache) FRCA provides a cache mechanism that dramatically improves the file serving performance of the HTTP Server (powered by Apache). Once a file is loaded into the below the MI cache called Network File Cache (NFC) the second request for that file can be served from the NFC. This eliminates open, read and close action for the file and is entirely handled below the MI by specially written SLIC code. FRCA can handle both a static file caching and a dynamic reverse proxy caching.

# Notes



**Proxy caching-** The IBM HTTP Server for iSeries can be configured 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 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 data available 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.

**Reverse Proxy Caching - Apache server only.**

A reverse proxy is another common form of a proxy server and 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 lessen 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. One 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.

# Notes



**Triggered Cache Manager (TCM)**- 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 page in the IFS when the underlying data changes, thereby improving the performance of a Web site. TCM is not a cache - it is a cache manager.



# Server Features: Configuration



## GUI Configuration and administration

Configuration for server: AS14PROXY  
Server type: **Original**  
Configuration: **PROXY**  
Change Server Settings

Forms for configuration: PROXY

**Basic settings**

- CGI
- Create configuration
- Delete configuration
- Directories and Welcome
- Display configuration
- Error message customization
- High Availability
- Java servlets
- Languages and Encoding
- LDAP
- Logging
- Log Reporting
- Meta-information
- PICS Local
- PICS Third-Party
- Protection
- Proxy Settings
- Request Processing
- Security configuration
- System Management

**Basic**

Configuration: PROXY

Host name: as14proxy

Bind options:  Bind server to host IP address  
 Bind server to all local IP address

Default port: 1010

User: %%SERVER%

Look up host name of requesting clients

Enable case sensitive mapping rules

Server-side includes:

- Disable
- Enable for CGI scripts only
- Enable for files only
- Enable for both CGI scripts and files

**HTTP Server for iSeries**

Welcome Setup Manage TCM Related Links

Server: APACHE00 - Apache Server area: Global configuration

Stopped

- Tasks and Wizards
  - Create New HTTP Server
  - LDAP Configuration
  - Add a Directory to the Web
  - Servlet and JSP Enablement
- Server Properties
  - General Server Configuration
  - Container Management
  - Virtual Hosts
  - URL Mapping
- Request Processing
- HTTP Responses
- Content Settings
- Directory Handling

- Security
  - Dynamic Content and CGI
  - Logging
- Proxy
- System Resources
- FRCA

- ASF Tomcat Setup task
- ASF Tomcat Settings
- WebSphere Application Server

**General Server Configuration**

General Settings Welcome Pages Configuration Includes Advanced

Autostart: Global

Server root directory: /tcp/dir00

Configuration file: conf/httpd.conf

Document root: Browse

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

Add

OK Apply Cancel

# Notes



**GUI Configuration and administration-** You can configure and administrate HTTP server instances from Web browsers. To show the configuration and administration screen, type the following URL from a Web browser:

http://hostname:2001

These figures below show the configuration screens for the HTTP Server (original) and HTTP Server (powered by Apache), respectively. The configuration screens are different between the HTTP Server (original) and the HTTP Server (powered by Apache), but you can reach each screens from the same URL.

The HTTP Server (original) and the HTTP Server (powered by Apache) can coexist. That is, you may have zero, one or many original servers running at the same time you have zero, one or many HTTP Server (powered by Apache) servers running. The administration screen allows you to create and manage HTTP servers. This figure shows the GUI Administration screen.

The navigation menu in the left-hand frame provides the following choices:

- **Tasks and Wizards:** Allows you to create new HTTP and ASF Tomcat servers with a wizard. The wizard provides a streamlined process for quickly creating HTTP servers. You can also “migrate” or create a new HTTP Server (powered by Apache) based upon an existing HTTP Server (original) without changing the existing configuration.
- **Global Settings:** This section allows you to configure the global server attributes (the rules and configuration settings for all servers) such as autostart, number of threads to use, and specific mapping tables. See image to right.
- **Internet Users and Groups:** Allows you to define who can access resources on the server.
- **Search Engine Setup:** Allows you to setup the Webserver Search Engine.




The screenshot displays the 'HTTP Server for iSeries' administration interface. The main title is 'HTTP Server for iSeries' with a navigation bar containing 'Welcome', 'Setup', 'Manage', 'TCM', and 'Related Links'. The left-hand navigation menu is expanded to 'Global Settings', which includes 'Global Server Settings', 'Access Control Lists (original only)', 'Internet Users and Groups', and 'Search Engine Setup'. The 'Global Server Settings' page is active, showing various configuration options: 'Autostart' is set to 'No'; 'Number of threads' has a 'Minimum' of 10 and a 'Maximum' of 80; 'Coded character set identifier' is set to '00819'; and 'Server mapping tables' are configured with 'Outgoing EBCDIC/ASCII table' and 'Incoming ASCII/EBCDIC table' both set to '\*CCSID'. The page concludes with 'OK', 'Apply', and 'Cancel' buttons.

© 2003 IBM Corporation

# Server Features: Virtual Host



## Virtual hosts

-  Both IP based
-  Both Name based
-  pbA Mass dynamic

# Notes



**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 ISP services to clients.

**IP based-** The IP based virtual host uses the IP address to route the requests from the clients to each virtual host. In this implementation, you can use HTTP 1.0 for the client web browser. But at the same time, you have to configure multiple IP addresses on your iSeries.

**Name based-** The name based virtual host uses the domain name to handle the requests from the clients. Since this implementation always needs the domain name in the URL header, the client web browser must support HTTP 1.1.

**Mass dynamic-** 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 IP address and the contents of the Host: header into the pathname of the file that is used to satisfy the request.

# Server Features: Security



Both

## Basic authentication via

- OS/400 user profiles
- Validation lists
- LDAP server

Both

## SSL (and TLS)

- Server
- Client

# Notes



**Basic authentication-** Basic authentication is a very popular way to secure Web resources. You can protect Web resources by asking the user for a userid and password to gain access to these resources. The userid and password can be validated in one of three ways:

- **OS/400 User profile** - This requires that each user must have a system user profile.
- **Validation list** - This requires you to create a validation list that contains Internet users. You can create a validation list and Internet users through the Configuration and Administration forms (click Administration --> Internet Users).
- **LDAP server** - This requires that you configure a LDAP server with the user entries.

**SSL and TLS-** Secure Sockets Layer (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.

There are multiple versions of the SSL protocol defined. The latest version, Transport Layer Security (TLS) Version 1.0, provides an evolutionary upgrade from SSL Version 3.0.

Both the HTTP Server (original) and the HTTP Server (powered by Apache) support server authentication and client authentication.

With server authentication, the client will ensure that the server certificate is valid and that it is signed by a Certificate Authority which the client trusts.

With client authentication, the server will ensure that the client certificate is valid and that it is signed by a Certificate Authority which the server trusts.

# Server Features: Application Serving



**Both** WebSphere Application Server plug-in

**pbA** Apache Software Foundation's Jakarta Tomcat

## Domino plug-in

**pbA** Domino R6 provides a plug-in for the HTTP Server (powered by Apache) (supported at V5R2 and V5R1)

**o** Domino Release 5 provides a plug-in for the HTTP Server (original)

# Notes



**WebSphere Application Server plug-in-** The IBM HTTP Server for iSeries handles static content, CGI program invocations, and proprietary plug-ins. The run-time environment (WebSphere Application Server) 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.

**ASF Tomcat-** The HTTP Server (powered by Apache) includes an industry-standard Java servlet and JavaServer Pages (JSP) container engine 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.

Available with the HTTP Server for iSeries (5769-DG1) in V4R5 is ASF Tomcat at version 3.2.1. Available with the HTTP Server for iSeries (5722-DG1) at V5R1 and V5R2 is ASF Tomcat at version 3.2.4.

For more information refer to the iSeries HTTP server Web page  
<http://www.ibm.com/eserver/series/software/http>

**Domino plug-in-** Domino 6: a Domino plug-in allows the HTTP Server (powered by Apache) to access Domino documents. This plug-in will immediately support the HTTP Server (powered by Apache) at V5R2 of OS/400 (this support is also PTFed back to V5R1).

Domino 6 will not support the HTTP Server (original) via plug-in at that time.

For the latest information in this regard see <http://www.ibm.com/servers/eserver/series/domino/>.



# Web Application Server Comparison



	WAS 3.5 Standard and Advanced ( <b>not supported in V5R2</b> )	WAS 4.0 Single server and Advanced	Tomcat 3.2.4 (V5R1 and V5R2)
<b>Servlets</b>	2.2 plus IBM extensions	2.2 plus IBM extensions	2.2
<b>Java Server Pages (JSP)</b>	1.1	1.1	1.1
<b>Java Developer Kit (JDK)</b>	1.2 and 1.3	1.3	1.2 and 1.3
<b>Enterprise Java Beans (EJB)</b>	1.0 (Advanced version only)	1.1	Not supported
<b>eXtensible Markup Language (XML)</b>	Supported	Supported	Not provided directly as part of 5722-DG1 but available as part of the IBM Toolbox for Java.
<b>Connection pooling</b>	Supported	Supported	Not supported
<b>Session support</b>	Supported - as an IBM extension to Servlet 2.2 support	Supported - as an IBM extension to Servlet 2.2 support	Not supported Sessions however, as defined in servlet 2.2, are supported by ASF Tomcat.
<b>J2EE compliant</b>	No	Yes	No

**What about: WebSphere Application Server - Express, V5?**

# Notes



**Servlets-** WebSphere Application server includes full support for servlet specification 2.1 and 2.2 (3.5.2 and later versions). In addition WebSphere Application Server includes its own packages that extend and add to the Java Servlet API. Those extensions and additions make it easier to manage session state, create personalized Web pages, generate better servlet error reports, and access databases.

**Java Server Pages (JSP)-** WebSphere Application Server version 3.5 includes full support for levels .91, 1.0, and 1.1 of the Java Server Pages (JSP) specification. Version 4.0 supports JSP level 1.1, only.  
ASF Tomcat includes full support for level 1.1.

**Java Developer Kit (JDK) support-** WebSphere Application Server and ASF Tomcat require a compatible Java Development Kit (JDK). The iSeries server supports the installation of multiple JDK. The JDK support is installed as options of the iSeries Developer Kit for Java (5722-JV1) license program.

**Enterprise Java Beans (EJB)-** WebSphere Application Server Advanced Edition at version 3.5 implements the EJB Version 1.0 specification, with some 1.1 specification enhancements, particularly in the area of the finder helper methods. Version 4.0 supports the EJB 1.1 specification.  
ASF Tomcat does not include support for EJB. Note: For an open source EJB solution you could employ JBoss. Refer to <http://www.jboss.org/>

**eXtensible Markup Language (XML)-** The IBM Toolbox for Java and JTOpen (JTOpen is the open source version of Toolbox for Java) is a library of Java classes that give Java programs easy access to iSeries or AS/400 data and resources. It includes the XML4J parser (IBM XML parser), Program Call Markup Language (PCML), an XML dialect, to support Java programs calling AS/400 application programs and Panel Definition Markup Language (PDML), an XML dialect, to support GUI panel definition.  
Any Java application (running on any system including, of course, an iSeries server) can make use of the special classes that are provided with the IBM Toolbox for Java including those written as servlets over ASF Tomcat

Note: For an open source XML solution you could employ Apache Cocoon. Refer to <http://xml.apache.org/cocoon/>

# Notes



**Connection pooling-** Connection pooling allows you to manage a pool of relational database connections. This Data Source support allows you to connect and disconnect to a data server more efficiently by pooling connections and reusing them. That reduces the overhead of connecting and disconnecting. And lets you control the number of concurrent connections to a data server product. Data Source connects to a remote database, pulls in required data, saves it into a local cache, and disconnects.

The Data Source utility is not supported by ASF Tomcat.

Note: For an open source solution you could employ Apache Cocoon with the esql.xsl. Refer to <http://xml.apache.org/cocoon/>

**Session support-** A session is a connection between a client and a server where information is exchanged. Session support allows a Web application developer to maintain state information regarding a user's visit. HTTP alone does not recognize or maintain a user's state. HTTP treats each user request as discrete, independent entity. In order to hold the user's state information sessions are defined as part of the servlet 2.2 specification.

Sessions, as defined by servlet 2.2, are directly supported by ASF Tomcat.

WebSphere Application Server provides an IBM extension for tracking user sessions, the Session Manager. The service is provided in the form of IBM classes and packages. WebSphere Application Server uses some mechanisms to address this session affinity: cookie and URL rewriting.

The Session Manager is not supported by ASF Tomcat.

**J2EE compliant-** The Java(TM) 2 Platform Enterprise Edition defines the standard for developing multi-tiered enterprise applications.

WebSphere Application Server Version 4.0 is fully J2EE(TM) 1.2 compliant.

For more information about J2EE, see the Sun Java(TM) 2 Platform Enterprise Edition Web site .

<http://java.sun.com/j2ee/>

# Notes



WebSphere Application Server Express as been announced. When this product becomes available for the iSeries this table will need to be updated. Currently the web site at <http://www.ibm.com/software/webservers/appserv/express/> says:

## WebSphere Application Server - Express, V5

With all the tools necessary to create and run a simple dynamic Web site--in one tightly integrated and affordable package-- IBM WebSphere® Application Server - Express, V5 offers a cost effective, approachable on-ramp to e-business - a ready-to-go, out-of-the-box solution. Based on the latest Java™ and Web services standards, WebSphere Application Server - Express lets you convert static Web sites into dynamic Web sites by viewing and performing simple information updates in back-end databases - while also providing the ability to consume Web services and resources for integrating with packaged applications.

WebSphere Application Server - Express provides:

- Quick , easy-to-use wizard-driven installation.
- Integrated development environment (available separately as IBM WebSphere Studio Site Developer for Windows® ) offering a simplified programming model focusing on JavaScript™ and Tag Libraries.
- Support for the latest specifications for Java ServerPages™ and Java Servlets.
- Development environment complete with wizards and samples that can be used as a starting point, code repository or reference and educational guide to help developers through the process of building a dynamic Web site.
- One-click application assembly and deployment and near-zero maintenance to minimize administration requirements.
- Smooth migration to other WebSphere Application Server and IBM WebSphere Studio configurations when more advanced development and deployment capabilities are required.

Related to this is the fact that WAS V3.5 is not supported on V5R2. Considering the near-future availability of WAS Express V5 this 'window' is considered small enough as to not pose a problem for most of our customers. Special note: A special 'as is' PTF is available for WebSphere Application Server 3.5.4 that allows it to run on OS/400 V5R2M0. It is PTF SI05225. WAS at 3.5.4 is still unsupported at V5R2M0.

# Notes



More detailed information: The long-awaited WebSphere Application Server (WAS) -- Express version for the iSeries is on its way. Express is a full-function Web application server, except that it lacks support for Enterprise Java Beans (EJBs) and a few other of WAS V5's enterprise-class functions. The iSeries version contains lots of platform-specific goodies, most notably the IBM Telephone Directory application. It will be available February 21 for V5R2 and March 14 for V5R1.

IBM U.S. Announcement Letter: [http://www.ibm.link.ibm.com/usalets&parms=H\\_203-008](http://www.ibm.link.ibm.com/usalets&parms=H_203-008)

IBM Canada Announcement Letter: [http://www.ibm.link.ibm.com/canalets&parms=H\\_A03-0055](http://www.ibm.link.ibm.com/canalets&parms=H_A03-0055)

IBM EMEA Announcement Letter: [http://www.ibm.link.ibm.com/emealets&parms=H\\_ERIFZP030158](http://www.ibm.link.ibm.com/emealets&parms=H_ERIFZP030158)

IBM Asia Pacific Announcement Letter: [http://www.ibm.link.ibm.com/aplets&parms=H\\_AP031017](http://www.ibm.link.ibm.com/aplets&parms=H_AP031017)

# Notes



# Server Features: Miscellaneous



**Both** Webserver Search Engine and Webserver Search Engine Web Crawler

**Both** Web-based Distributed Authoring and Versioning (WebDAV)

◦ Access log reporting and Web usage mining

◦ Platform for Internet Content Selection (PICS)

**pbA** Collection Services

**Both** Log Rollover and Maintenance

**pbA** Header Control (expires, etc.)

**pbA** URL rewriting

**Both** NLS translated GUI

# Notes



**Webserver Search Engine and Webserver Search Engine Web Crawler-** The HTTP Server search engine allows you to perform full text searches on 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 (5769-DG1 or 5722-DG1) starting at OS/400 V4R4. You can control what options are available to the user and how the search results are displayed through customizable Net.Data macros.

Some of the features of the Webserver Search Engine include:

- Indexes documents for fast searching:
  - The iSeries Webserver Search Engine indexes HTML or text files into a format that allows a large number of documents to be searched quickly. Multiple indexes can be created, and documents from multiple directories can be placed in a single index.
- Exact word indexing:
  - The Webserver Search Engine uses an exact word indexing scheme rather than a keyword indexing scheme used by many search engines. All words are indexed; nothing is left out. Exact word indexing provides for faster index building and more precise searching than keyword indexing, but requires additional disk space. Documents are searched using consecutive character matching, which is essential for proper support of double-byte languages.
- Advanced search functions:
  - The AS/400 Webserver Search Engine supports advanced search capabilities such as exact search, fuzzy search, wild card search, proximity search, English word stemming, case-sensitive search, boolean search, and document ranking.
- Customizable search forms:
  - The search forms and search results form are completely customizable by the end user using the Net.Data scripting language. This gives the user the ability to specify the type of search to be done and how the results are to be displayed. The information that can optionally be displayed on the results page includes the number of documents satisfying the search, number of occurrences of the search term, number of documents returned on this page, the URL associated with each document, the document's ranking, and the last modified date and size. Any and all of this information can be displayed however the user chooses.
- Web-based administration:
  - Administration of the search indexes is handled as part of the IBM HTTP Server Configuration and Administration Web pages. The search administration forms allow you to create and delete search indexes, update search indexes when documents are modified, and view the status of an index.
- Multiple language support:
  - The Webserver Search Engine supports multiple national languages including double-byte languages Chinese, Japanese, and Korean.

On the iSeries the search engine comes in two logical pieces that are Webserver Search Engine and Webserver Search Engine Web Crawler. And they are related to each other.



# Notes



**Web-based Distributed Authoring and Versioning (WebDAV)-** 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:

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, refer to: <http://www.webdav.org/>

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

**Platform for Internet Content Selection (PICS)-** PICS support enables labels (metadata) to be associated with Internet content. Originally designed to help parents and teachers control what children access on the Internet, it also facilitates other uses for labels, including code signing and privacy.

# Notes



**Collection Services-** is adding support in V5R2 for user defined categories and probes. This support will allow IBM products and customer applications to capture application unique performance data along with the system data already provided by collection services.

The IBM HTTP Server (powered by Apache) is going to use this support to provide performance data specific to the server. This will not be done with the original server.

**Log Rollover-** The Original IBM HTTP server 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 will not create any logs - the proper directives must be configured by the web administrator in order 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 will extend the HTTP Server (powered by Apache) code to include log rollover support. This will be in the form of the HTTP Server (original) support, and then extended by allowing the user to specify one of the following values: "Off" "Hourly", "Daily", "Weekly", "Monthly". The directive that provides log rollover support is "LogCycle". If a daily log cycle is desired, that would be achieved with by placing the following directive in the configuration file:

```
LogCycle Daily
```

Values of "Daily", "Weekly" and "Monthly" will cause the logs to rollover at midnight of the respective time period. A value of "Hourly" will cause the logs to rollover at the top of each hour".

Note: log file formats are not changing in V5R2.

# Notes



**and Maintenance-** in V5R2 for the HTTP Server (powered by Apache) provides log maintenance support similar to the HTTP Server (original). This will provide log maintenance mechanisms similar to those for HTTP server, including deletion of files based on both age and aggregate size variables. Log Maintenance within the Apache Server will happen at midnight. The maintenance job will be started up each midnight by the main server (after a timer pops) and see whether any maintenance needs to be done. If maintenance is required it will be performed at this time. If it is not required, the maintenance job will end.

The Original HTTP Server has the ability to remove log files after the logs reach a certain size or age. This is controlled through server configuration directives.

Apache currently does not have a built-in capability to do log removal. Traditionally on Apache, archiving was done by shutting Apache down, moving the logs elsewhere, and then restarting Apache. There is a program called `rotatelog`, which can be found in the Apache support subdirectory. It closes the log periodically and starts a new one, and is useful for long-term archiving and log processing.

We plan to improve on the support provided by the Original HTTP Server by generalizing and expanding some of the existing directives so that they may be used more generically for multiple log files, rather than using specific directives for each active log file.

Log maintenance can be configured with the new `LogMaint` directive.

When log maintenance is done on a file, it is purged from the system. Candidates are determined by looking at the appropriate variations of the filename given on the `LogMaint` directive. These variations will have extensions and names containing the format `Qcyyymmddhh`. The server will generate names of this format as long as the log cycle function is active via the `LogCycle` directive.

Log files may be purged when they reach a user-defined age in days or when the overall size of log files exceeds a user-defined limit. Values for both size and age can be specified on the `LogMaint` directive.

Files that are currently in use (currently open) by the server instance are not eligible to be removed.

If non-zero values have been configured for both age and size limit, files which have exceeded the age limit are deleted first. After aged files have been deleted, the server calculates the size of all remaining files. If the user-defined size limit is still exceeded, the server will continue to purge log files until either the aggregate size of remaining log files is within the size limit or all eligible files have been deleted.

# Notes



**URL rewriting:** allows the rewriting of URLs to URLs, URLs to filenames, and filenames to filenames. This uses the mod\_rewrite engine that is part of every Apache server. For example say that you want to prevent a particular user agent called Webcrawler from accessing any pages on the server. To do this you would include the following directives in your configuration:

```
RewriteEngine on
RewriteCond %{HTTP_USER_AGENT} ^Webcrawler
RewriteRule ^.*$ - [F,L]
```

The first line enables the rewrite engine. The second line provides a test that returns true if the HTTP\_USER\_AGENT string starts with the letters Webcrawler. If the second line is true, then the third line takes any URL string and returns a forbidden message to the client.

**Header Control (expires, etc.):** The HTTP Responses form provides tabs that allow you to configure an HTTP server for maintaining how server error messages are handled, configuring response header attributes, specifying the expiration parameters for cached files, and configuring meta file processing.

## HTTP Responses ?

**Error Message Customization** | **Response Headers**

**Expires Header** | Meta-information Files

Generate Expires HTTP header for responses:  ?

Default expiration time: ?

Expiration time:

After:

Set expiration time based on resource MIME type: ?

	MIME file type	Time	After
Example	image/gif	2 Days	Document access by client

Corporation

# Notes



**NLS translated GUI:** The V5R2 HTTP Server (powered by Apache) has been fully translated for all 51 languages. PTFs should be coming soon to roll similar MRI PTFs into V5R1 near the months of Sept or October 2002.

# Notes



## Not on the list:

**Translation and serving of OS/400 Spoolfiles:** Neither the original nor the HTTP Server (powered by Apache) can directly serve OS/400 spoolfiles.



# Getting Started

## Get the Redbook!

### HTTP Server (powered by Apache), SG24-6716

- Redbook, published April-3-2002, last updated April-10-2002

### Know the Web site:

- [www.iseries.ibm.com/http](http://www.iseries.ibm.com/http)

### Signup for 'Friends of...'

- During the break

### Update the Redbook!

- Six week residency: IS-3302
- Starts April 28th; Rochester
- Goto: [ibm.com/redbooks](http://ibm.com/redbooks)

F03EP01v5r2ipbA.PRZ

The image shows the cover of an IBM Redbook titled "HTTP Server (powered by Apache)". The cover features the IBM logo in the top right corner. The title is prominently displayed in large, bold, black font. Below the title, it says "An Integrated Solution for IBM @server iSeries Servers". The cover includes three bullet points: "Go from zero to your first HTTP Server (powered by Apache) in minutes", "Fully exploit the integrated power of OS/400 and Apache", and "Go step-by-step through the Web site and applications". A large graphic of a globe is shown, with a thick black line forming a stylized 'G' shape around it. The authors' names are listed at the bottom right: Brian R. Smith, Gaia Banchelli, Monica Maria Echeverry, and Axel Lachmann. The Redbooks logo is at the bottom right, and the URL [ibm.com/redbooks](http://ibm.com/redbooks) is at the bottom left.

IBM

# HTTP Server (powered by Apache)

An Integrated Solution for IBM @server iSeries Servers

- Go from zero to your first HTTP Server (powered by Apache) in minutes
- Fully exploit the integrated power of OS/400 and Apache
- Go step-by-step through the Web site and applications

Brian R. Smith  
Gaia Banchelli  
Monica Maria Echeverry  
Axel Lachmann

**Redbooks**

[ibm.com/redbooks](http://ibm.com/redbooks)



# Software



<b>Mandatory</b>	<b>Product or option number</b>
IBM HTTP Server for iSeries	5722-DG1
TCP/IP Conectivity Utilities	5722-TC1
Java Developer Kit 1.2	5722-JV1 *Base and option 3
<b>Recommended</b>	
WebSphere development ToolSet	5722-WDS and option 51 (Compiler ILE C)
Triggered Cache Manager	Option 1 of 5722-DG1
<b>Security</b>	
Digital Certificate Manager (DCM)	Option 34 of OS/400
Cryptographic Access Provider	5722-AC2 or 5722-AC3
<b>Servlet &amp; JSP Programming</b>	
WebSphere Application Server Standard Edition or WebSphere Application Server Advanced Edition	5733-AS3 or 5733-WA3
Toolbox for Java	5722-JC1
Java Developer Kit	5722-JV1 options
Qshell Interpreter	option 30 of OS/400

# Notes



Mandatory		
IBM HTTP Server for iSeries	5722-DG1	The LPP is IBM HTTP Server for iSeries, which contains the component HTTP Server (powered by Apache).
TCP/IP Connectivity Utilities	5722-TC1	Very useful collection of TCP/IP applications including Telnet, FTP and others.
Java Developer Kit 1.2	5722-JV1 *Base and option 3	Your HTTP Server (powered by Apache) requires this LPP.
Recommended (but optional)		
WebSphere development ToolSet	5722-WDS and option 51 (Compiler ILE C)	Needed if you will be doing any application development in ILE languages.
Triggered Cache Manager	Option 1 of 5722-DG1	Needed only if you will be working with Triggered Cache Manager (TCM).
Security (optional)		
Digital Certificate Manager (DCM)	Option 34 of OS/400	Optional: to support the handling of digital certificates used by SSL/TLS for secure Web serving.
Cryptographic Access Provider	5722-AC2 or 5722-AC3	If you want to use SSL or TLS you must install one of the IBM Cryptographic Access Provider products. The availability of these products is subject to USA export regulations and can order them as a separate no-charge LPP.

# Notes



Servlet & JSP Programming (optional)		
WebSphere Application Server Standard Edition or WebSphere Application Server Advanced Edition	5733-AS3 or 5733-WA3	The Standard Edition with its support of servlets and JSPs at version 3.5 is all you really need to compare and contrast with the Apache Software Foundation's Jakarta Tomcat.
Toolbox for Java	5722-JC1	If you will be programming in Java this is recommended.
Java Developer Kit	5722-JV1 options	Your HTTP Server (powered by Apache) requires this LPP *Base and Option 3 (Java Developer Kit (JDK) 1.2). If you will be programming in Java you might need some other options for other levels of the JDK.
Qshell Interpreter	option 30 of OS/400	Useful for working with WebSphere Application Server.

# Notes



# Components and PTFs



## Components of 5722-DG1

Name of component	Option number
HTTP Server (original)	*base
HTTP Server (powered by Apache)	*base
Net.Data	*base
AS/400 Webserver Search Engine	*base
Apache Software Foundation's Jakarta Tomcat	*base
Highly Available HTTP Server	*base
Triggered Cache Manager	Option 1 of 5722-DG1

## Group PTFs

Product	PTF description
IBM HTTP Server for iSeries (5722-DG1) (V5R2)	Group PTF SF99098
IBM HTTP Server for iSeries (5722-DG1) (V5R1)	Group PTF SF99156
IBM HTTP Server for iSeries (5769-DG1) (V4R5)	Group PTF SF99035

Goto: <http://www.ibm.com/servers/eserver/series/software/http/> and click PTFs

# Notes



LPP IBM HTTP Server for iSeries (5722-DG1) contains these components. That is, when you install 5722-DG1 on your iSeries server you will also be able to make use of:

Name of component	Option number	Comment
HTTP Server (original)	*base	
HTTP Server (powered by Apache)	*base	in V4R5 as group PTF SF99035
Net.Data	*base	
AS/400 Webserver Search Engine	*base	
Apache Software Foundation's Jakarta Tomcat	*base	A Java servlet and JSP application server
Highly Available HTTP Server	*base	A V5R1 enhancement only
Triggered Cache Manager	Option 1 of 5722-DG1	

Since the HTTP Server (powered by Apache) is so new and ever-changing it is mandatory that you install the latest and greatest fixes for IBM HTTP Server for iSeries (5722-DG1) and other related products. Table below shows us the products and PTFs that you must install on your iSeries server:

Product	PTF description	Comments
IBM HTTP Server for iSeries (5722-DG1) (V5R1)	Group PTF SF99156	The current version of this V5R1 group PTF can be displayed on the iSeries by running the following command: DSPDTAARA QHTTTPSVR/SF99156
IBM HTTP Server for iSeries (5769-DG1) (V4R5)	Group PTF SF99035	The current version of this V4R5 group PTF can be displayed on the iSeries by running the following command: DSPDTAARA QHTTTPSVR/SF99035

# User Profile Authorities



## Your user profile must have

- \*IOSYSCFG authority
- \*CHANGE authority to the library QUSRSYS

## Objects require \*ALL authority

- QUSRSYS/QATMHINSTA
- QUSRSYS/QATMHINSTC

## Commands require \*USE authority

- CRTVLDL
- STRTCPSVR
- ENDTCPSPVR

## Web browser Needs to support

- HTTP 1.0 or 1.1 protocol
- Frames
- Java Script

# Notes



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

Tip: Do not use QSECOFR as this user profile will not work with the HTTP server's Configuration and the Administration forms. You can, however, use a user profile that has a user class of \*SECOFR.

- 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
- ENDTCPVSR

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. The security of the log files should be fully considered. The path of the log files should only be accessible by the appropriate user profiles.



# Notes



The 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

Browsers such as Microsoft's Internet Explorer 5.5 or later, and Netscape Navigator 4.75 will work with the Configuration and Administration forms.

**Tip:** Due to some cookie caching problems, we advise you to use the Microsoft's Internet Explorer.

In order to view the log reports generated by the HTTP Server, you must use a browser which supports JVM 1.1.5 or later.

# Notes



# Test the Installation of HTTP Server

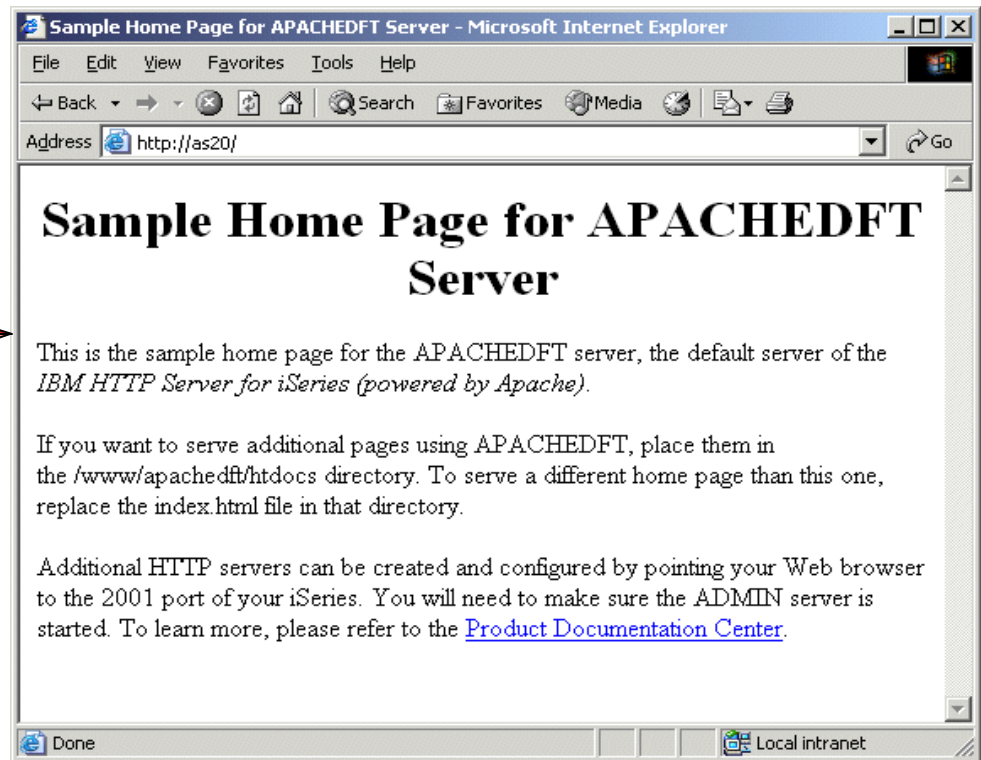


- Using iSeries Navigator
  - Start Instance -> APACHEDFT

or

- Type command
  - STRTCPSVR SERVER(\*HTTP) HTTPSVR(APACHEDFT)

- From your Web browser
  - `http://your.server.name/`



**Do not copy directives in  
APACHEDFT  
to your own server configuration!**

# Notes



One very good and quick way to see if your iSeries server has all the pieces to run an HTTP Server (powered by Apache) is to start the APACHEDFT server that comes with the installation of IBM HTTP Server for iSeries (5722-DG1).

**Tip:** Do not use (for example, by copy and paste) the configuration directives found in the APACHEDFT server for your own server. The reason is that the Port directive used within has been deprecated. The directive Listen should be used instead.

The best way to create a new Web server configuration is via the wizard.

To test your installation, do the following:

1. Verify that the iSeries TCP/IP is started.
2. Verify that the host name and the domain name for your iSeries server is specified and matches an entry in the local host table.
3. Verify that you will not have any IP address and port conflicts with other HTTP Server (original) and HTTP Server (powered by Apache) Web servers. This is important as the APACHEDFT server is configured to bind to all IP addresses and TCP/IP port 80 which is the default port for Web applications.

4. Start the APACHEDFT server instance

Using Operations Navigator expand Network -> Servers -> TCP/IP and right-click HTTP Administration. Then click Start Instance -> APACHEDFT.

Or, from the iSeries command line:

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

This will start the APACHEDFT server instance listening on all active IP interfaces on TCP/IP port 80. Starting this server instance will only allow read access to files located in the directory /www/apachedft/htdocs (and all sub-directories).

5. Open your Web browser and enter;

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

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

If everything is correctly setup and configured you should see the default welcome page (physically located in the iSeries IFS at /www/apachedft/htdocs/index.html) as shown above.

# Apache Configuration GUI



## Enhanced GUI for configuration using the "Admin" server

- Port 2001 (http://myserver:2001) for iSeries Tasks page
- Wizard based configuration for common and complicated tasks
- Start/stop via GUI

IBM®  
(C) IBM Corporation 2000

## iSeries Tasks

as20.itsoroch.ibm.com

- [IBM HTTP Server for iSeries](#)  
Configure the iSeries HTTP Server and SSL
- [IBM WebSphere Commerce Suite, Pro Edition for iSeries](#)  
Configure the Commerce Suite server
- [Digital Certificate Manager](#)  
Create, distribute, and manage Digital Certificates
- [IBM IPP Server for iSeries](#)  
Configure the IBM IPP Server
- [4758 Cryptographic Coprocessor](#)  
Configure the 4758 coprocessor

HTTP Server with WebSphere Application Server - Express for iSeries

Welcome Setup Manage TCM Related Links

### Set Up Your Servers

Set up the servers that define your web presence, set up how users will access your servers, and define how searching will be done across your web site.

**Tasks and Wizards**  
Select to create a HTTP server, or migrate an existing original HTTP server to a new HTTP (powered by Apache) server, or create an application server.

**Global Settings**  
Change your global settings that will affect all of your HTTP servers, as well as set up your access control lists for use with your original servers.

**Internet Users and Groups**  
Set up your Internet users and groups for additional access control to your web presence. Certificates can be used to identify and group users.

**Search Engine Setup**  
Set up a search engine for your web site. This allows you to control how searching will be done across your web servers.

# Notes



**Enhanced graphical configuration & administration** - Enhanced Graphical User Interface for configuration using the "Admin" server. Wizard based configuration for common and complicated tasks. Start and stop server via GUI. Simplifies the configuration Process.

The "look and feel" of the iSeries Tasks page is the same from V4R5 to V5R2. So far nothing really has changed.

# GUI Setup



## HTTP Server Creation:

- Create new HTTP, WAS Express and Tomcat servers; migrate Original to Apache

## Global Settings:

- Global attributes

## Internet Users and Groups:

- Define who can access server resources

## Search Engine Setup

HTTP Server with WebSphere. Application Server - Express for iSeries

Welcome Setup Manage TCM Related Links

Tasks and Wizards

- Create New HTTP Server
- Create New Express Server
- Migrate Original Server to Apache
- Create ASF Tomcat Server

Global Settings

- Global Server Settings
- Access Control Lists (original only)

Internet Users and Groups

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

Search Engine Setup

### Set Up Your Servers

Set up the servers that define your web presence, set up how users will access your servers, and define how searching will be done across your web site.

#### Tasks and Wizards

Select to create a HTTP server, or migrate an existing original HTTP server to a new HTTP (powered by Apache) server, or create an application server.

#### Global Settings

Change your global settings that will affect all of your HTTP servers, as well as set up your access control lists for use with your original servers.

#### Internet Users and Groups

Set up your Internet users and groups for additional access control to your web presence. Certificates can be used to identify and group users.

#### Search Engine Setup

Set up a search engine for your web site. This allows you to control how searching will be done across your web servers.

# Notes



## Setup

The setup options allow you to create and manage HTTP servers. The navigation menu in the left-hand frame provides the following choices:

**Tasks and Wizards:** Allows you to create new HTTP, WAS Express, and ASF Tomcat servers with a wizard. The wizard provides a streamlined process for quickly creating HTTP servers. You can also “migrate” or create a new HTTP Server (powered by Apache) based upon an existing HTTP Server (original) without changing the existing configuration.

**Internet Users and Groups:** Allows you to define who can access resources on the server.

**Search Engine Setup:** Allows you to setup the Webserver Search Engine.



# Notes



**Global Settings:** This section allows you to configure the global server attributes (the rules and configuration settings for all servers) such as autostart, number of threads to use, and specific mapping tables.

Autostart: specifies if all servers should automatically start when TCP/IP starts.

Number of threads: specifies the minimum and maximum number of threads to use for all servers.

Minimum: specifies the minimum number of threads the servers keep open. From 1 to 9999 threads can be specified. \*DFT specifies that the default value of 10 is used. This parameter is not used by HTTP Server (powered by Apache).

Maximum: specifies the maximum number of threads the servers open. From 1 to 9999 threads can be specified. \*NOMAX specifies that there is no limit to the number of threads the server can open. \*DFT specifies that the default value of 40 is used.

Coded character set identifier: specifies the five digit number from 00001 through 65533 that represents the coded character set identifier (CCSID) all servers should use. The servers use the CCSID to translate documents from EBCDIC to ASCII. The servers use this value when a character set and page are not identified in the MIME header from the web browser. \*DFT specifies that the default value of 00819 is used. For a list of CCSID values see the Globalization topic in the iSeries Information Center.

Server mapping tables: specifies the tables that you want the servers to use to convert EBCDIC to ASCII and ASCII to EBCDIC.

Outgoing EBCDIC/ASCII table: specifies the table that you want the servers to use for EBCDIC to ASCII character conversion for outgoing documents. \*CCSID specifies the CCSID value for determining outgoing mapping. \*GLOBAL specifies that the outgoing mapping defined in the Global Server settings is used.

Library: specifies the library name where the table is located. The library name is required when specifying a specific outgoing table. The library name is not a required when \*GLOBAL or \*CCSID is specified for the outgoing table.

Incoming ASCII/EBCDIC table: specifies the table that you want the servers to use for ASCII to EBCDIC character conversion for incoming documents. \*CCSID specifies the CCSID value for determining incoming mapping. \*GLOBAL specifies that the incoming mapping defined in the Global Server settings is used.

Library: specifies the library name where the table is located. The library name is required when specifying a specific incoming table. The library name is not a required when \*GLOBAL or \*CCSID is specified for the incoming table.

# Notes



**Access Control Administration:** allows you to define who can access resources on your server.

A **validation list** is an AS/400 object of type \*VLDL that stores user names and passwords for use in access control. Validation lists are case-sensitive. Validation lists reside in AS/400 libraries and are required when adding 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.

Enter an existing **Group file** directory with a fully-qualified path name in the integrated file system 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 will be 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 Integrated File System.

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 might, 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 will be 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 will be added directly into that group.

## Add Internet user

User name:	<input type="text"/>
Password:	<input type="text"/>
Confirm password:	<input type="text"/>
Comments:	<input type="text"/>
Validation list:	<input type="text"/>
Group file:	<input type="text"/>
Group:	<input type="text"/>

# Notes



At a high level, these are some of the things you could say about the GUI administration:

- Enhanced Graphical User Interface for configuration using the "Admin" server
- Wizard based configuration for common and complicated tasks
- Migration of original HTTP server configurations to Apache via Wizard
- Start/Stop/Restart server instances
- Display status of all instances
- Written in Java, runs as a servlet
- Provides default directives which make configurations "secure by default" and adds common directives you will want
- Provides API's for programmatic access and updates to configuration files
- Simplifies the configuration process

# Notes



# HTTP Server Creation



- Create new Original or Apache
- Merged path with migration of Original to Apache wizard
  - either explicit down a Migrate from Original to Apache wizard
  - or Create a 'new' Apache based upon an Original

\*Migrate Original Server to Apache

\*Create New HTTP Server

### Create New HTTP Server

Welcome to the Create New HTTP Server wizard. This wizard helps you set up and create new HTTP servers. You may exit the wizard at any time without making any changes by clicking Cancel.

*IBM HTTP Server for iSeries* gives you two types of servers:

- HTTP server (powered by Apache), which is based on the Apache Software Foundation's popular web server implementation. This is the strategic server for *IBM HTTP Server for iSeries*.
- HTTP server (original), which is the same web server that has been available since V3R2 and is based on a CERN web server implementation.

Which type of server do you want to create?

HTTP server (powered by Apache) - **recommended**

HTTP server (original)

Back Next Cancel

© 2003 IBM Corporation

# Notes



**Merged path with migration of Original to Apache wizard:** Either by selecting to migrate the Original HTTP server to Apache directly, or by creating a new Apache server and asking that it be based upon an Original HTTP server configuration. Upper left panel shows the panel during the Create HTTP server (Apache) wizard that allows you to base it upon an Original configuration. Lower right is the first panel of the Migrate Original Server to Apache wizard.

## Create New HTTP Server

Would you like to create your new server based on the configuration of an existing server?

No

Yes, create based on an existing original HTTP server

HTTP server (original) ASPHTTP

**Note:** This involves a migration of the original server configuration to an Apache configuration. This in no way alters or destroys the existing server or its configuration.

## Migrate Original Server to Apache

Welcome to the Migrate Original Server to Apache wizard. This wizard helps you create a new HTTP server (powered by Apache) based on an existing HTTP server (original) or configuration. You may exit the wizard at any time without making any changes by clicking Cancel.

Do you want to base your new server configuration on an existing original server or an original named configuration?

HTTP server (original) ASPHTTP

Original named configuration

**Note:** This involves a migration of the original server configuration to an Apache configuration. This in no way alters or destroys the existing server or its configuration.

# HTTP Server Creation Worksheet



**Here are the variables you must know to complete the wizard:**

Create HTTP Server wizard question	Options	Answer example
HTTP server type	HTTP server (powered by Apache) or HTTP server (original)	HTTP Server (powered by Apache)
HTTP server name	name	ITSO88
Migrate Original server configuration?	(Yes/No) If Yes, name the Original HTTP server instance	No
Server root	default: /www/webserver	/ITSO/ITSO88
Document root	default: /www/webserver/htdocs	/ITSO/ITSO88/ITSOco
Listen on IP address and Port	default: All IP addresses default: Port 80	All IP addresses 8088
Logging	Do you want your new server to use an access log?	Yes. Note: The Apache server always has an error log.

# Notes



**HTTP Server type:** The IBM HTTP Server for iSeries gives you the ability to create two different types of web servers

- The HTTP server (powered by Apache) which is based on the popular Apache web server implementation. This is the strategic code base for the IBM HTTP Server for iSeries.
- The HTTP server (original) which is the same web server that has been available since V3R2 and is based on a CERN web server implementation.

**HTTP server name:** This name will be used later to configure and administer your server. The name must begin with an alphabetic character. The name can have up to nine additional alphanumeric characters. It may not contain any blanks or special characters.

**Migrate Original server configuration:** Yes, configure based on an existing Original server. Note: This involves a migration of the original server configuration to an Apache configuration. This in no way alters or destroys the existing server or its configuration.

**Server root:** The server root is the base directory for your HTTP server. Within this directory, the wizard will create subdirectories for your logs, and configuration information. Note: If the server root does not exist, the wizard will create it for you.

**Document root:** The document root is the directory from which your documents will be served by your HTTP server. Note: If the document root does not exist, the wizard will create it for you.

**Listen on IP address and Port:** Your server may listen for requests on specific IP addresses or on all IP addresses of the system. Note: Most browsers make requests on port 80 by default. The wizard gives you a pull-down of valid IP addresses already configured on your iSeries.

**Logging:** Your HTTP server can keep a log of activity on your site. The HTTP Server (powered by Apache) will always have an error log (by default). If you specify Yes, a second activity log will be created and will contain information on access requests, and both the Referrer and UserAgent headers from incoming requests.



# Notes



This is the configuration file generated by the parameters in the worksheet.

```
# Configuration originally created by Apache Setup Wizard Thu Jan 30 16:42:20 UTC 2003
Listen *:8088
DocumentRoot /itso/itso88/itsoco
ServerRoot /itso/itso88
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 /itso/itso88/itsoco>
  Order Allow,Deny
  Allow From all
</Directory>
```

# Notes



# Migration Wizard



**V5R2 enhanced to now migrate additional original server directives:**

- Highly Available
- Log rollover and archiving
- Proxy
- Protection setups
- Virtual hosting
- Server Side Include (SSI)

**Adding a new request routing directive, Map, to eliminate ordering problems with request routing differences**

**Also adding a decision point**

- Do you want the migration wizard to migrate request routing directives

# Notes



**Adding a new request routing directive, Map:** By using a specially coded rewrite rule via mod\_rewrite we can finally properly migrate the original Map directive to something that works the same in Apache. This will eliminate ordering problems with request routing differences. Map was one of the last request routing directives from the Original server to now be migrated to the HTTP Server (powered by Apache).

## HTTP Server News

At <http://www.ibm.com/servers/eserver/series/software/http/news/sitenews.html> the Rochester Apache developers are looking for your source original configuration files to help verify the changes they have made to the migration wizard:

May 20, 2002 IBM Rochester Accepting Sample Configuration Files

IBM Rochester would like some real world, working server configuration files for both the "original" IBM HTTP Server and IBM HTTP Server (powered by Apache) to determine whether they can offer more complex coverage for their migration wizard and GUI interfaces. IBM may benefit in helping improve their test suites and you may benefit in improved quality. There may be some commonalities across the configuration files that could potentially result in future articles or product enhancements if IBM sees some common concerns in multiple user configuration files.

If you would like to participate, send an email to [rchapach@us.ibm.com](mailto:rchapach@us.ibm.com) stating you wish to participate. You will receive a return email explaining the terms and conditions under which IBM can accept your configuration file.

# Manage: Servers



## Select Manage tab -> All servers (or specific server)

- Select server from list
- List can be sorted by column
- Stop, Start, Restart, Delete, Rename - depending on status of server
- Jump straight to configuration with Manage Details
- At bottom of list (not shown) are for the optional startup parameters

HTTP Server with WebSphere Application Server - Express for iSeries

Welcome Setup **Manage** TCM Related Links

Server: All servers

Tasks and Wizards

- Create New HTTP Server
- Create New Express Server
- Migrate Original Server to Apache
- Create ASF Tomcat Server

### Manage All Servers

Data current as of 04:47:33 PM UTC on 01/30/2003

Server	Type	Status	Autostart	Address:Port
ADMIN	Apache	Running	Global	*:2001 *:2010
APACHE98	Apache	Running	Global	*:59800
APACHEDFT	Apache	Stopped	Global	*:80
DEFAULT	Original	Stopped	Global	as20:80
ENGINE1	ASF Tomcat	Stopped	N/A	9.5.92.14:8009
forum01	WAS - Express V5	Stopped	N/A	*:10002,10007,10011,10012,10013
FORUM1HTTP	Apache	Stopped	Global	*:10001
FRCA99	Apache	Stopped	Global	*:8599
FRCATEST	Apache	Stopped	Global	9.5.92.28:80
GEORGEW	Apache	Running	Global	*:57700
INSURANCE	Original	Stopped	Global	9.5.92.28:80

Refresh Start Stop Restart Monitor

Manage Details Delete Rename

# Notes



## Manage HTTP Servers

The Manage HTTP Servers form allows you to view the current status, start, stop, restart, rename, monitor, delete, and configure your servers. Select the server that you want to work with by clicking the radio button to the left of the server name.

To view the current status of a server, find the server that you want to work with under the HTTP Server column of the table. The Type column specifies if the server is a HTTP Server (original) or a HTTP Server (powered by Apache) type. The Status column indicates if the server is running or stopped.

The Refresh button allows you to refresh the status.

To start or restart a server do the following:

- Select the server that you want to start or restart.
- Optionally specify any startup parameters in the Server startup parameters: field.

The following parameters are valid for the HTTP Server (powered by Apache):

- -netccsid [nnn] Overrides the DefaultNetCCSID directive
- -fscsid [nnn] Overrides the default DefaultFsCCSID directive
- -d [serverroot] Set the initial value for the ServerRoot variable to serverroot. This can be overridden by the ServerRoot directive.
- -f [configuration] Use the values in the configuration on startup. If the configuration does not begin with a /, then it is treated as a path relative to the ServerRoot.
- -C [directive] Process the given directive just as if it had been part of the configuration.
- -c [directive] Process the given directive after reading all the regular configuration files.
- -vv Enable verbose level service tracing. The -vv parm is ignored on a restart and does not toggle on/off tracing.
- -vi Enable informational level service tracing.
- -ve Enable error level service tracing.
- -V Display the base version of the server, build date, and a list of compile time settings, then exit.
- -l Display a list of all modules compiled.
- -t Test the configuration file syntax but do not start the server. This command checks to see if all DocumentRoot entries exist and are directories.

# Notes



Click the Start or Restart button. Starting the server issues the STRTCPSVR CL command. Restarting the server issues the STRTCPSVR CL command with the RESTART parameter.

To stop a server do the following:

- Select the server that you want to stop. Click the Stop button. Stopping the server issues the ENDTCPSPVR CL command.

To rename a server do the following:

- Select the server that you want to rename. Click the Rename button.

To monitor a server do the following:

- Make sure that you have enabled monitoring in the server configuration that you want to monitor. Select the server that you want to monitor.

Click the Monitor button. You will see a display of several server statistics such as the number of active threads.

To delete a server do the following:

- Select the server that you want to delete. Click the Delete button. For HTTP Server (powered by Apache) the server is deleted, but the server root directory and anything in or under that directory (like the configuration file) is not deleted. For HTTP Server (original) the server is deleted, but the associated server configuration is not deleted.

To change the configuration of a server do the following:

- Select the server that you want to configure. Click the Configure button.

# Notes







# A Quick Guide to Apache Contexts and Request Routing

# Notes



This part shows you the basic concepts of configuration and request routing with your HTTP Server (powered by Apache). We will then show you how to apply what you've learned by stepping you through a simple configuration scenario using the administration GUI.

- ▶ In-context configuration
- ▶ How context works
- ▶ Apache server request routing
- ▶ Apache request processing
- ▶ Configuration recommendations

# In-context Configuration



A mechanism to subset the configuration file into logical entities

- The entire configuration is considered to be the "Server Config" context
- Other contexts provide ways to subset your configuration to have different behavior based on the "context" that directive is located in.
- Contexts include:
  - Directory - uses the physical path the URL of the request maps to: `<Directory /mydir/>`
  - Files - within Directory context, allows further definition based on physical file names: `<Files abc.html>`
  - Location - uses the URL of the request: `<Location /www/>`
  - VirtualHost - uses the virtual host of the request: `<VirtualHost 1.2.30.40>`
- Allows you to do your custom configuration
- Some directives only work in the "Server Config"
  - Look at the directive documentation to learn if it is valid within a context

# Notes



You can think of context as containers of settings and it allows us to divide the configuration file into some logical entries.

The entire server configuration is itself considered to be a context called the global context or the server config context. Other contexts can have their own set of directives to behave differently. We can use several types of context as follows:

- Directory
- Directory Match
- Files
- Files Match
- Include
- LDAP Include
- Limit
- Limit Except
- Location
- Location Match
- Virtual Host

Each context must be defined in a certain upper level context. For example, Directory context must be defined in server config or virtual hosts context. Virtual host context can be contained only in server config context.

The contexts that you'll use most often are Directory and Files. Directory contexts define configuration settings for an entire directory, and Files contexts define settings for files matching a particular name pattern.

Other context types serve special purposes, and you'll probably need to 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 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.

# How Context Works



URL from Web browser

http://as20/downloads/downloads.html

HTTP server : ITSOCO

...

Document root /itso/itso99/itsoco

...

<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

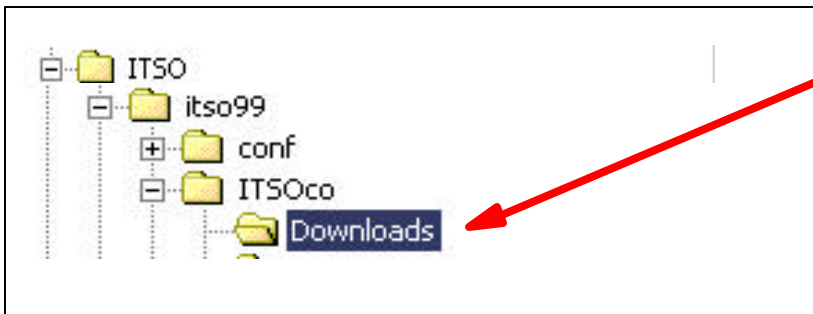
</Directory>

<Directory /itso/itso99/itsoco>

order allow,deny

allow from all

</Directory>



# Notes



If you are familiar with the HTTP Server (original) you'll 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 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, the thinking in the Apache world is that a simpler approach to configuration reduces errors that might otherwise compromise security.

Because it's unlikely that you'd 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 of the server ITSOCO might look like this:

ITSOCO global settings

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 subcontexts 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.

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.

# Apache Server Request Routing



URL mapping to file name is done with `mod_alias` directives

- This is configuration file order driven
- Includes `Alias(Match)`, `ScriptAlias(Match)`, and `Redirect(Match | Temp | Permanent)` directives

After `mod_rewrite` and `mod_alias`, "Directory walk" is done

- This step builds up the directives by "walking" the contexts which are hierarchically processed (presented from low to high)
  1. Server Config (overridden by `.htaccess` and other containers)
  2. Directory
  3. File
  4. Location
  5. Sections in Virtual Hosts are applied after corresponding sections outside.
- Sub-directories and sub-locations inherit directive settings
- Directives found in more specific contexts override
- Directives in the last container overwrite those in the first

# Notes



Both the HTTP Server (original) and the new HTTP Server (powered by Apache) filter 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 (original) processes directives sequentially; when a certain Pass or Exec matches the incoming URL, the configuration directive is applied and rule processing stops.

Similarly, 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 residing outside the `DocumentRoot`; for example, you could use `ScriptAlias` to map `/cgi-bin/` to `/QSYS.LIB/yourlib.lib/db2www.pgm`. Note that, by default, Apache configuration is not case sensitive.

However, most Apache request processing uses a procedure called a directory walk, in which 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`, then those within `/itso/itso99/itsoco/downloads`, all the while merging the settings of the directives that it finds. This is a more powerful mechanism than the HTTP Server (original) `Pass/Exec` syntax because 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's pattern-matching characters.
2. `DirectoryMatch` contexts and `Directory` contexts containing 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 contexts that occur earlier higher-level contexts. Or they can override those settings. Directives that apply to subdirectories can override those for parent directories. If there are two sets of directives that apply to the same directories, the last set read may override the others.



# Apache Request Processing (1 of 4)




STEP 1:

<http://as20/itso/itso99/itsoco/downloads/downloads.html>



- Server reads first part of URL - "/"
- This matches the first `<Directory />` container in the configuration.
- There are no other "/" directives in the file, so the conditions in this container apply to "/" and all of its subdirectories.
- The directive `deny from all` protects all directories from any accesses.



```
<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>
```

# Notes



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 is to allow access, but this is overridden by the next directive, `deny from all`. At the top of this configuration hierarchy we allow no access unless we override at a lower level. This approach secures the server by default.

# Apache Request Processing (2 of 4)



STEP 2:

<http://as20/itso/itso99/itsoco/downloads/downloads.html>

- Next some portions of the URL are read. This matches the `<Directory /itso/itso99/itsoco>` container.
- The directives in this container **allow** access to everything in the `"/itso/itso99/itsoco"` directory and all of its subdirectories.

```
<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>
```



# Notes



Next, the request routing process examines Directory /itso/itso99/itsoco. We want to allow open access to DocumentRoot (which contains our ITSOco server's home page), so 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.

# Apache Request Processing (3 of 4)



STEP 2:

<http://as20/itso/itso99/itsoco/downloads/downloads.html>

- Finally, the rest of the URL is read. This matches the `<Directory>` container that has the longest directory definition.
- The directives in this container will override the settings to deny access from all clients except `10.10.0.0/255.255.0.0`.

```
<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>
```



# Notes



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 will ignore the final three directives in this 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".

**Tips:** In order to process requests, web servers must decipher the URL sent in the browser's request, in order to determine which resource is being requested. The server does this by comparing the URL to the directives in the configuration file(s).

The AS/400's original HTTP server reads the URL, and then steps through the configuration file a directive at a time, until it finds one that matches the URL. Once it finds the first match, the original server processes the request and goes no further in the configuration file.

The Apache server also reads the URL and compares it to the configuration file. But, it does not stop at the first match. Instead, it looks for all matches, and prioritizes them from least specific to most specific ("shortest directory component to longest").

# Apache Request Processing (4 of 4)



Here are the directives that apply to the file "downloads.html" after each step:

## STEP 1:

**/itso/itso99/itsoco/downloads/downloads.html**

```
AllowOverride None
order deny,allow
deny from all
```

## STEP 2:

**/itso/itso99/itsoco/downloads/downloads.html**

```
AllowOverride None
order deny,allow
deny from all
order allow,deny
allow from all
```

## STEP 3:

**/itso/itso99/itsoco/downloads/downloads.html**

```
AllowOverride None
order deny,allow
deny from all
order allow,deny
allow from all
order deny,allow
allow from 10.10.0.0/255.255.0.0
deny from all
```

```
<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>
```

# Notes



This chart summarizes the way that directives are applied and then overridden.

The 'order' directive is sometime confusing to new Apache students. It is worth spending the time reading the Apache text books, web sites, or help via the iSeries Apache server to understand how it works ahead of time - because the students \*will\* ask!

If the configuration is:

```
order deny,allow
deny from all
allow from somehost
```

This means (in order of precedence):

- 1) allow is the default and would rule the day if no deny or allow was specified.
- 2) To deny, you must explicitly specify the clients you wish to deny. In this example, we are saying we want to deny all clients (by IP address)
- 3) Lastly, allow will override the denied clients. In this case the client coming from the IP address behind somehost would be the only client allowed to this directory.



# Configuration Recommendations



- **Do not use <Location> sections at all (unless truly necessary)**
  - Almost everything can be done with <Directory> sections.
- **Use <Files> sections only when you really need them.**
  - In most cases, a solution can be found by putting all these files into a separate directory and using a <Directory> section.
- **Keep the number of sections in the configuration file at minimum.**
  - Sometimes rearrangement of the directory structure helps.

## **Use .htaccess files only when you need to have distributed administration and configuration.**

- Avoid using several .htaccess files in one directory path (for example, /www/.htaccess and /www/html/.htaccess).

# Notes



Nested contexts and configuration directives can be confusing. Here are a few simple 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 - but you can use Directory sections for almost everything you need to do.
- Use Files 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.

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`).

You can avoid using `.htaccess` file by putting `Options FollowSymLinks` directive in a `<Directory>` container like:

```
<Directory />  
  Options FollowSymLinks  
  AllowOverride None  
</Directory>
```



# Virtual Hosts

# Notes



In this part, we will show you the concept of virtual host, and describe three different ways of implementation.

- ▶ Virtual host concept
- ▶ One server vs multiple server
- ▶ Virtual host implementation
- ▶ IP based implementation
- ▶ IP based: Problem scenario
- ▶ IP based: Solution
- ▶ Name based implementation
- ▶ Name based: Problem scenario
- ▶ Name based: Solution
- ▶ Mass dynamic implementation
- ▶ Mass dynamic: Problem scenario
- ▶ Mass dynamic: Solution

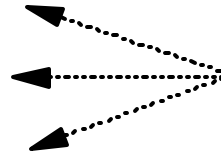
# Virtual Host Concept



iSeries

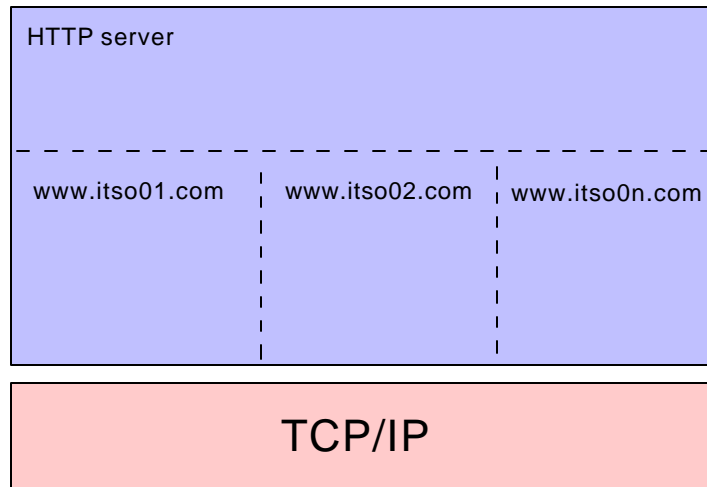


www.itso01.com  
www.itso02.com  
www.itso03.com

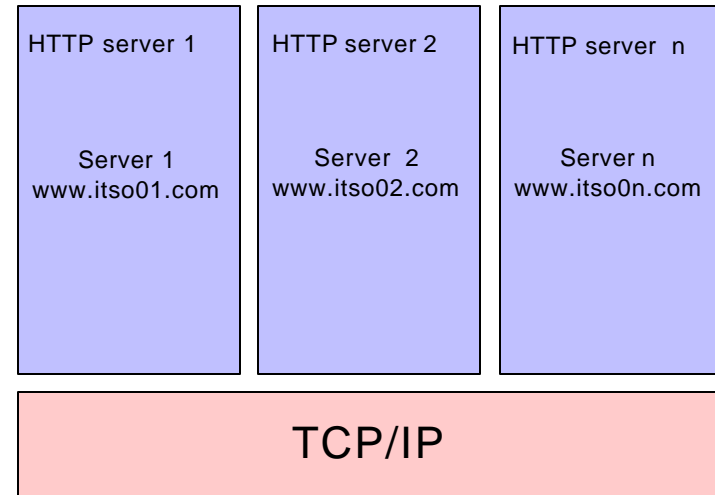


Client with web browser

One HTTP server



Multiple HTTP servers



# Notes



The concept of virtual host in terms of Web serving, refers to the practice of maintaining more than one domain in a single server. The way the domains are primary differentiated is by their host name or IP address. Thus, client request are routed to the correct domain by IP address or by host name contained in the URL header. Traditionally the virtual host implementation requires as many HTTP servers running simultaneously as domains the system is going to serve.

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 us to use one HTTP server to host as many domains as the environment requires.

The virtual host concept is primary used by ISPs (Internet Server Provider), content providers or companies who 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 a ISP to host and publish their Web pages. The ISP then setup the virtual host implementation in a way that each site looks like runs in a different server. Each one of this servers is called a virtual host as they are running on the same server.

The HTTP server can be configure to host:

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

# One Server vs Multiple Server



## Configuration comparison

- Number of files and directories
  - Configuration file
  - Log files
  - Document root directories
- IP address or domain name to serve the request
- Directives to handle each domain

## Environment comparison

- Configuration file, process, service directives
- Number of processes
- User profile to run the domains
- Impact of failure

**Both can work properly, but less resources for "one server"**

# Notes



The HTTP configuration is different if the domains are served using one or from multiple HTTP servers, as each approach requires different process and is based on different server directives, as explained in the table below.

<b>One HTTP server</b>	<b>Multiple HTTP servers</b>
One configuration file shared by all the domains.	Multiple configuration files, one for each domain.
One or many access log and error log files.	Each configuration file include 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 the each domain.	No specific server directives as each domain has his own configuration file.



# Notes



The runtime environment of each approach also differs, as illustrated in this table.

<b>One HTTP Server</b>	<b>Multiple HTTP Servers</b>
All domains run under the same environment: configuration file, process and some service directives.	Each domain runs under his own environment: configuration file, process and server directives.
One process is started as there is only one HTTP server.	Many process are started as there are more than one HTTP server.
All the domains run under the same user profile, as there is only one server process. However each one can be configured under a different security mechanism. Refer to Chapter 5, "Defending the IFS" on page 89 for more information.	Each domain can run under his own user profile, as there are one process per domain. You can change the user profile the server instance runs on, using the ServerUserID directive.
If you have problems with one domain, and the recovery procedure implies the HTTP server restart, all the domains serve by the HTTP will be restarted.	If you have problems with one domain, the HTTP server associated with the domain can be restarted without affect any other domain.

The information provide on these tables allow us to identify the differences between the two approaches. According to those tables, both HTTP server process the visitor request correctly. Usually deciding between running under one or multiple HTTP servers depends more of the system resources, like memory and CPU, and security issues, like independency between domains, than of any HTTP server directive limitation that could be used with any approach.

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

# Notes



# Virtual Host Implementation



## IP based

- Uses the IP address to serve the domain
- Client Web browser can be HTTP 1.0
- Needs multiple IP addresses configured

## Name based

- Uses the domain name in the URL header to serve
- Can share a single IP address
- Client Web browser must support HTTP 1.1

## Mass dynamic based

- Uses the domain name to serve the request
- Can share the contexts among multiple domains

# Notes



**IP based-** The HTTP server uses the IP address to handle the visitors request. Can be used with HTTP 1.0.

**Name based-** The HTTP server uses the domain name include into the URL header to handle the visitors request. This implementation requires HTTP 1.1.

**Mass dynamic based-** The HTTP Server retrieves the domain name provide in the URL header to process the data requested by the client. The differences here is that this is done dynamically, which means, the domains does not have to be registered using any <VirtualHost> context in the HTTP configuration.

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:

- How many 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 as there are HTTP version 1.0 and HTTP version 1.1. The HTTP Server (powered by Apache) and most of the Web client browsers support HTTP 1.1 protocol. The differences between the protocols are:
  - With the HTTP 1.0 protocol, the HTTP server relies on the 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 include in the visitor request (as a header) thus the HTTP server receives the domain name and can process the request according to the HTTP directives include into the configuration file.

As not all the client browsers support the HTTP version 1.1 protocol, the HTTP server must be configure in a way that no matter what limitations the environment has, each client request will be handle correctly. To accomplish this, the HTTP Server (powered by Apache) supports three different virtual host implementations.

When you configure the IP based virtual hosts, your iSeries server must have multiple IP addresses in one of two ways.

- Multiple IP addresses over one physical connection
- Multiple IP addresses over multiple physical connections

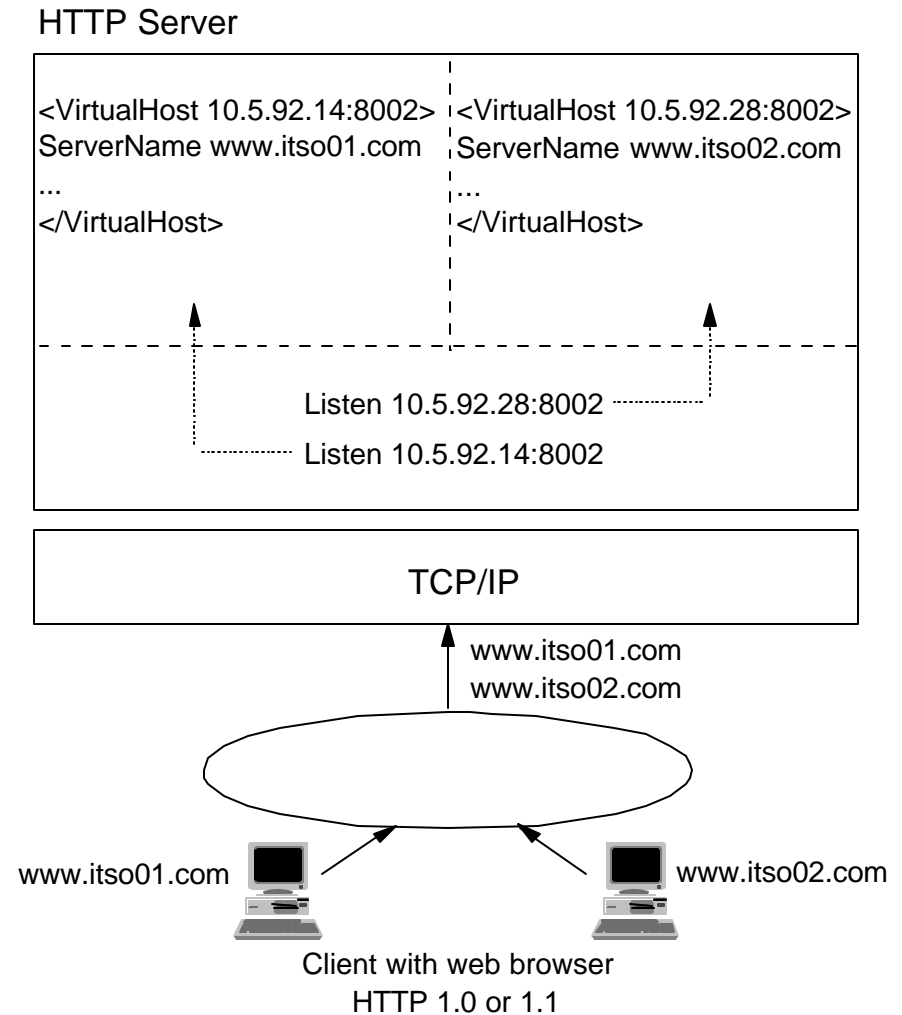
Your iSeries server can be configured as a multi-homed server with multiple IP interfaces or virtual IP addresses. Any of those IP addresses can be used with the HTTP server to handle multiple domains in one iSeries server.

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

# IP Based Implementation



- Different IP address for each virtual host
- Web browser can be HTTP 1.0
- Good approach for domains in different network



# Notes



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. This can be done by either having multiple physical network connections or having multiple IP addresses. The TCP/IP implementation on the iSeries server support multiple IP addresses implementation.

Once you have identified the IP address for each domain, you just tell the HTTP Server (powered by Apache) how to handle it using the <VirtualHost> directive. The IP based implementation works very well but requires a dedicated IP address for every virtual host the system is going to serve and usually this means more cost especially if you must purchase these additional IP addresses and domains from an Internet Service Provider (ISP).

The IP based virtual host implementation provide an immediate solution for any browser as the implementation does not rely on any specific browser functionality and therefore 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 show in the figure.

The IP based implementation supports multiple domains using a different IP addresses. This is a good implementation approach if you want to have each domain running in a different network using his own IP address.

# IP Based: Problem Scenario



## Situation

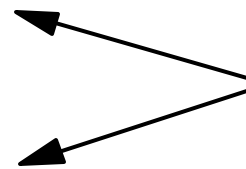
- Need to host two different domains
- Web browsers do not all support HTTP 1.1

iSeries



www.itso01.com  
10.5.92.14:8002

www.itso02.com  
10.5.92.28:8002



Client with browser  
HTTP 1.0

# Notes



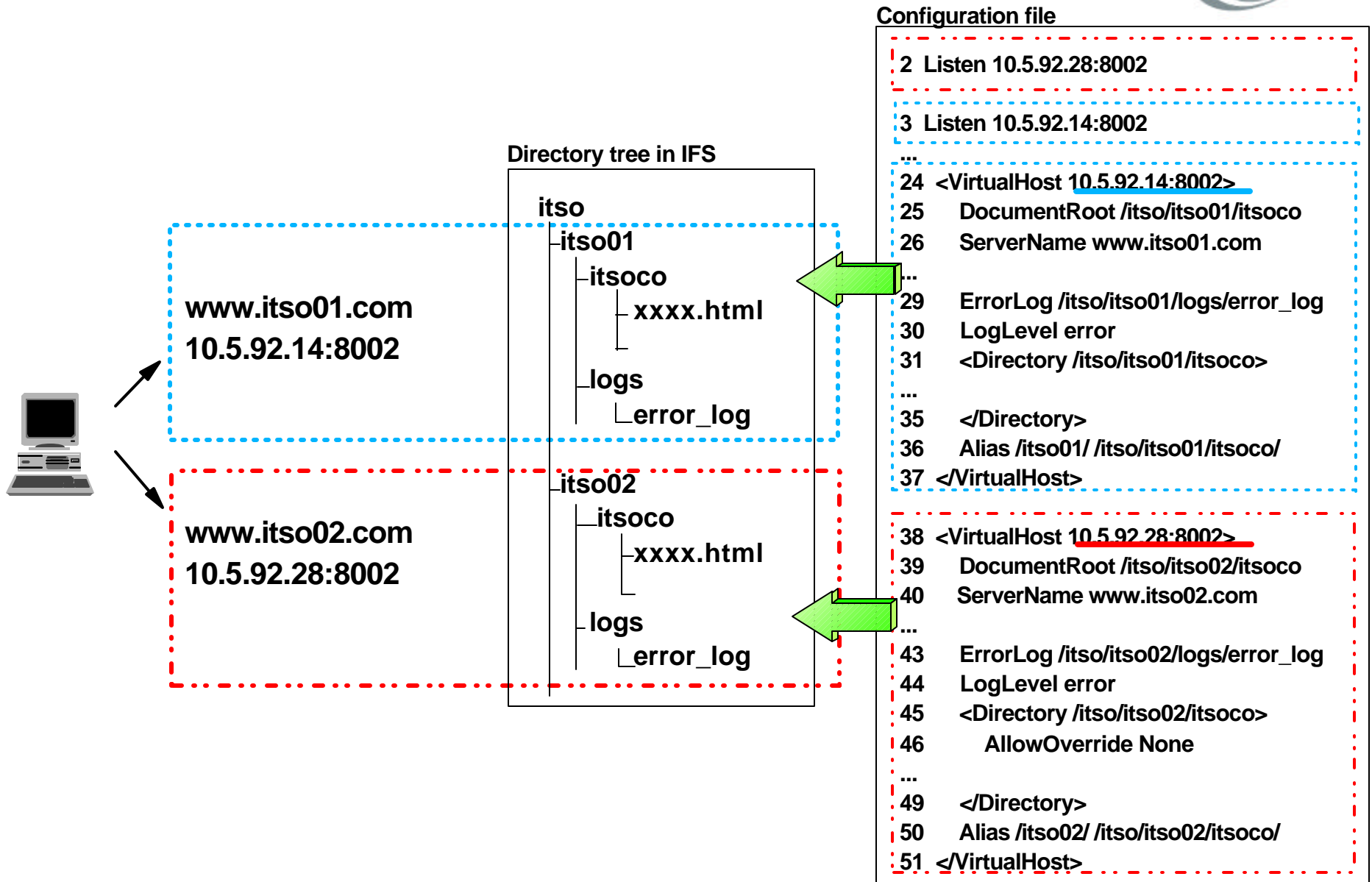
Your company needs to host two different domains, `www.itso01.com` and `www.itso02.com` using one iSeries server. As your iSeries server has two available IP addresses and the Web browser clients do not all support the HTTP 1.1 protocol, you have decided to create an IP based virtual host implementation.

In order to configure the HTTP Server (powered by Apache) to handle the client requests, we have to identify some of the basic resources used by the HTTP server to route and serve those domains. The following table shows some of the basic resources used by HTTP server.

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



# IP Based: Solution



# Notes



**Listen** - This directive tells the server to listen for request on more than one IP address or port. In this example, we define 2 different IP addresses for each virtual host.

**<VirtualHost>**- This directive introduces a virtual host context. When you use IP based virtual host, you should set different IP address in this directive for each virtual host. In this example, we have 2 virtual host contexts for serving 2 domains. VirtualHost context ends with the directive **</VirtualHost>**.

VirtualHost context includes directives that provide individual environment for each virtual host. These directive includes DocumentRoot, ServerName, ErrorLog, <Directory>, and so on.

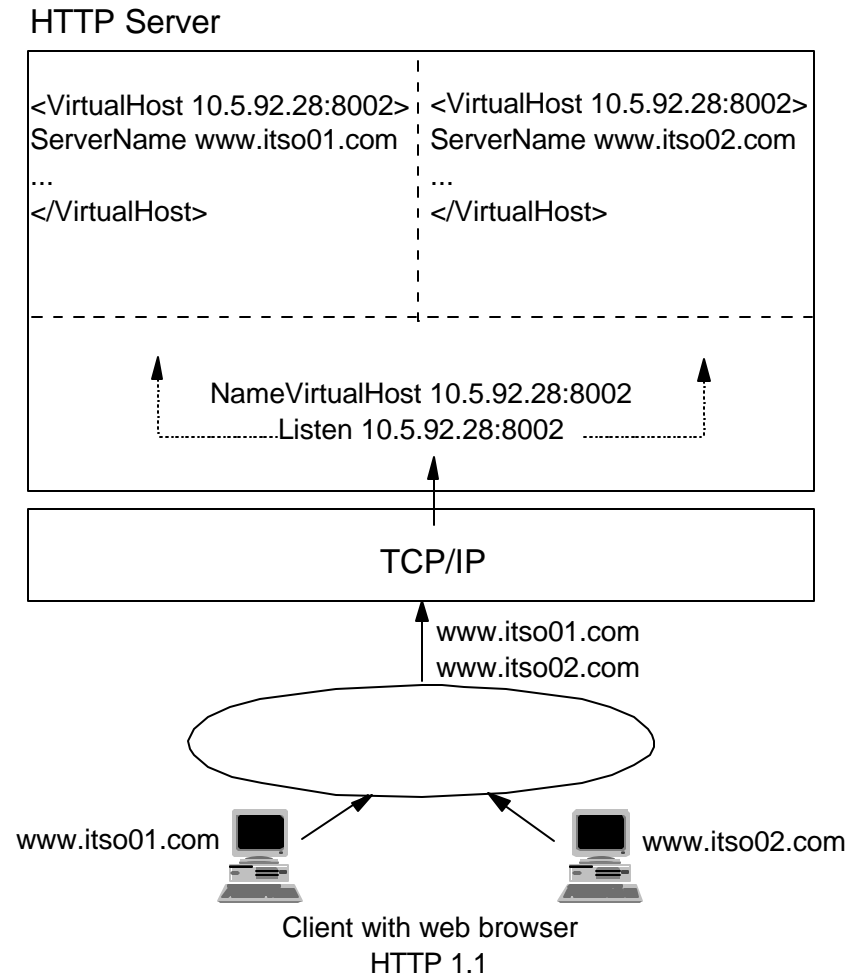
- **DocumentRoot**- This directive sets the directory from which the virtual host serves files. In this example, one domain uses /itso/itso01/itsoco and the other one uses /itso/itso02/itsoco directory.
- **ServerName**- This directive sets the host name of the server and used when creating redirection URLs. In this example, one domain has www.itso01.com, and the other one has www.itso02.com as the hostname.
- **ErrorLog**- This directive sets the name of the file to which the server will log any errors it may encounter.
- **LogLevel**- This directive adjusts the verbosity of the messages recorded in the error logs.
- **<Directory>**- The directory context can be included in the virtual host context and take effect inside the virtual host.
- **Alias**- This directive allows documents to be stored in the local filesystem other than under the DocumentRoot.

```
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 ...
```

# Name Based Implementation



- No additional IP addresses needed
- Simple configuration
- Web browsers must support HTTP 1.1



# Notes



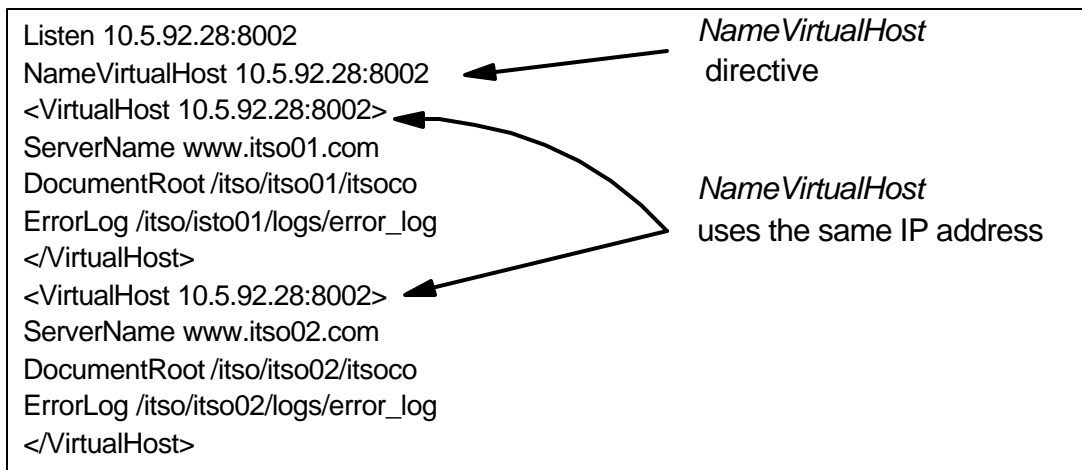
The named based virtual host implementation allows one IP address and TCP/IP port to host more than one domain. The benefits of using the name based virtual hosts implementation is a practically unlimited domains, ease of configuration and use, and it requires no additional hardware or software resources.

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

Simply put, name based virtual hosting requires that the clients request, which are being routed to the same physical interface with the same IP address, carry the hostname in the HTTP headers so the HTTP server can distinguish between virtual hosts.

Besides the `<VirtualHost>` directive used by IP based implementation, the name based uses the `NameVirtualHost` directive. This directive specifies an IP address (or hostname that is mapped to an IP address) that should be used as a target for name based virtual hosts as shown on the figure below. Although `www.itso.com` can be hostname it is recommended that you always use an IP address for performance reasons. Any additional directive can (and should) be placed into the `<VirtualHost>` context. The HTTP Server configuration name based will look like the following:

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



# Name Based: Problem Scenario

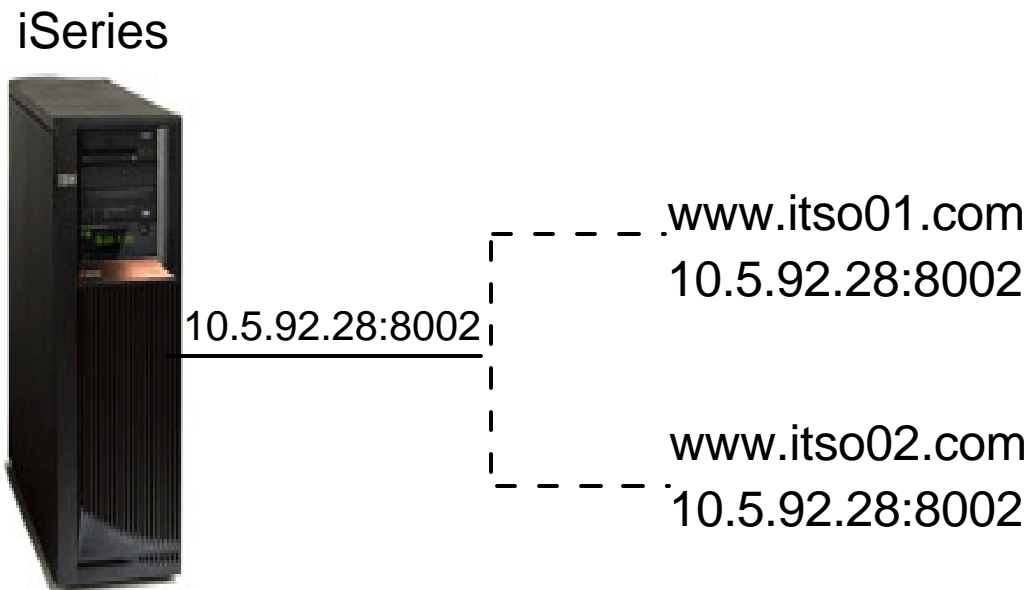


## Situation

- Need to host two different domains
- Only one IP address is allowed

## Consideration

- All Web browser clients must support HTTP 1.1



# Notes

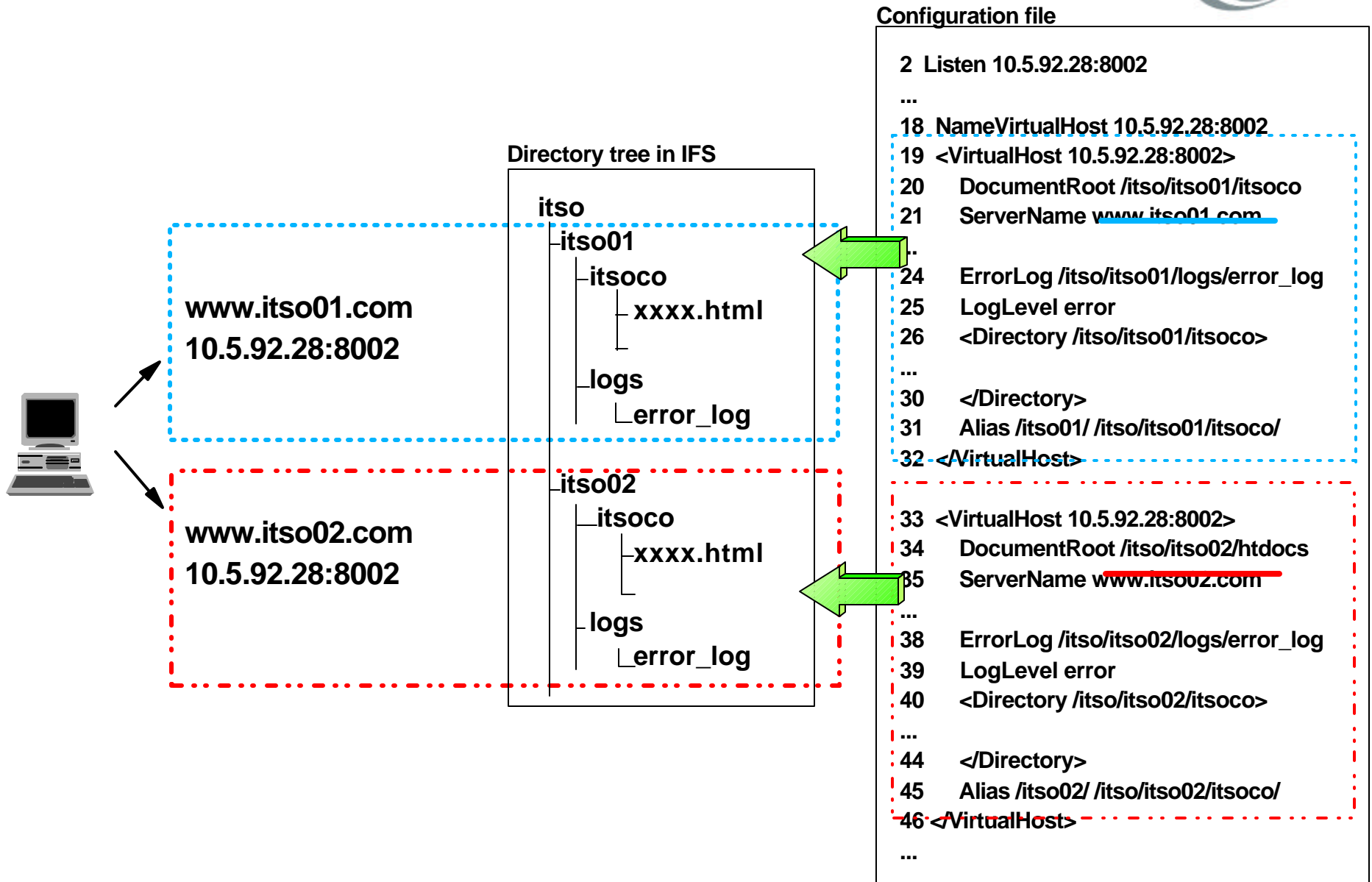


This time, your company needs to host two different domains, `www.itso01.com` and `www.itso02.com` sites but your iSeries server only has one IP address to serve the incoming requests. As the system only has one IP address, we decided to serve both domains using the name based virtual host implementation. This will allow the HTTP server to handle the client request based on the domain name as shown in the figure. Note, all the Web browser clients must support the HTTP 1.1 protocol.

In order to configure the HTTP Server (powered by Apache) to handle the visitors request, we have to identify information related to the domain configuration as shown in the table below. With that information we can create the name virtual host configuration for our site. The Web server resources are the same as the one we used for the IP based implementation but now your iSeries server only has one IP address for all incoming client requests.

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

# Name Based: Solution



# Notes



**Listen** - For name based virtual host, you don't need to have multiple IP addresses. In this example, we specify this single IP address.

**NameVirtualHost**- This directive is a required directive if you want to configure name based virtual hosts.

The notable difference between IP based and name-based virtual host configurations is this NameVirtualHost directive. The directive specifies an IP address and should be used as a target for name-based virtual hosts.

Although hostname can be specified in this directive, it is recommended that you always use an IP address.

When your request arrives at this IP address, matching <VirtualHost> context is searched and the one hostname matches ServerName is selected.

**<VirtualHost>**- For name based virtual host, every <VirtualHost> directive has the same IP address.

You can specify several directives inside the virtual host context in the same way as IP based virtual host.

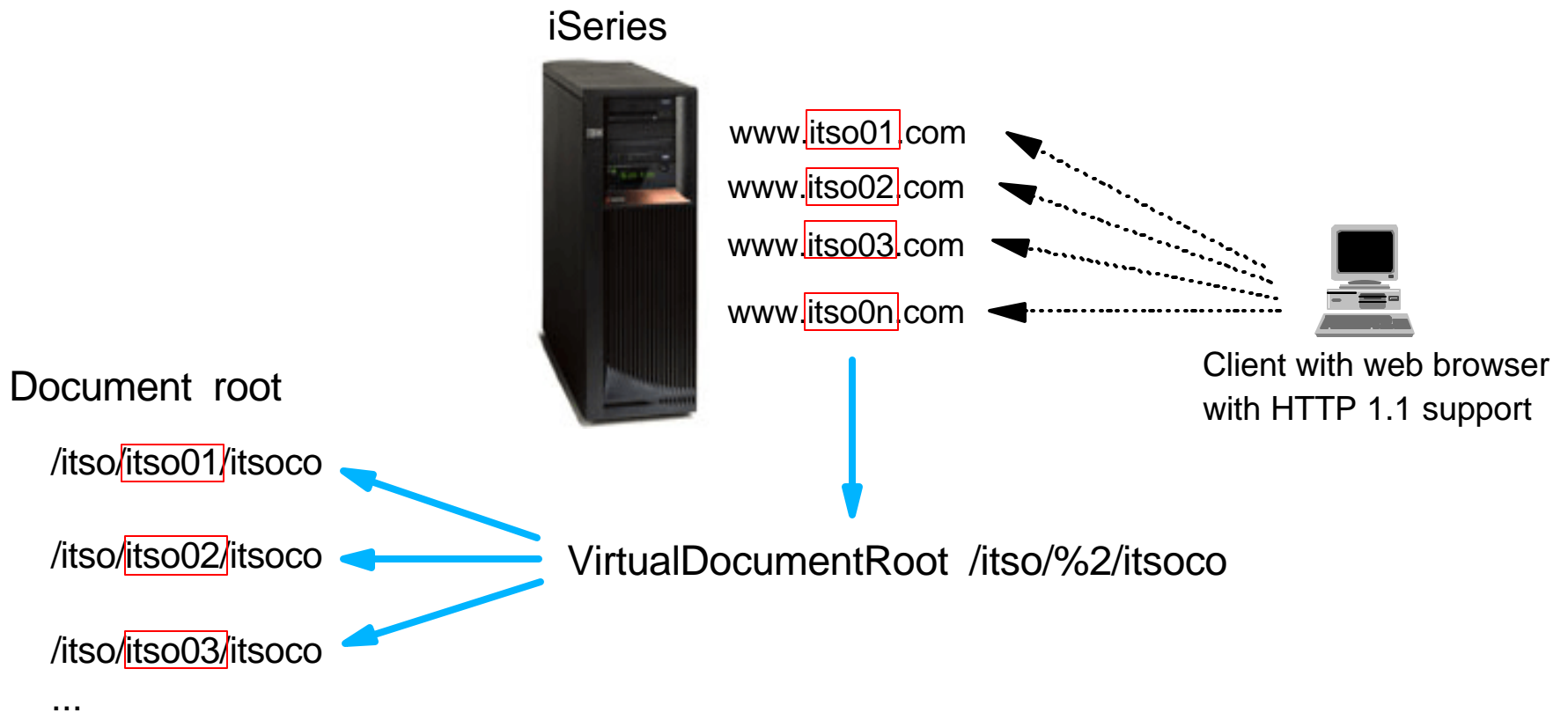
```
1 # Configuration originally created by
2 Listen 10.5.92.28:8002
...
18 NameVirtualHost 10.5.92.28:8002
19 <VirtualHost 10.5.92.28:8002>
20     DocumentRoot /itso/itso01/itsoco
21     ServerName www.itso01.com
22     UseCanonicalName Off
23     HostNameLookups off
24     ErrorLog /itso/itso01/logs/error_log
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>
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>
```



# Mass Dynamic Implementation



- Dynamically changes the document root
- Host name in URL is used
- Eliminates many <VirtualHost> sections



# Notes



The mass dynamic virtual host implementation allows us 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, using the host name provide in the URL requested by the client, the HTTP server process the request as show in the figure.

The mass dynamic virtual host implementation differs from the IP based or the name based in the mechanism used to determine the location of the files you want to serve. Here, the HTTP server uses the content of Host provide in the URL to serve the visitors request. Basically, the mass dynamic virtual host uses like a variable path name (based on the header) to find into the file system structure the static data the site is going to server. 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 this table.

Variable	Value
<code>%%</code>	insert a %
<code>%p</code>	insert the port number of the virtual host
<code>%N.M</code>	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.

# Notes



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 this table.

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
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 4-11 and the values in the Table 4-12, the mass dynamic performs the interpretation process, called Directory name interpolation. The interpolation process requires that the interpolated directory exists into the file system as the web server name will be 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 will fail. The 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 look for the document root based on the value of the server name.
- `VirtualDocumentRootIP` allows you to determine where the server look for the document root based on the IP address.
- `VirtualScriptAlias` allows you to specify the directory path where the server look for CGI scripts based on the value of the server name.
- `VirtualScripAliasIP` allows you to specify the directory path where the server look for CGI scripts based on the IP address.

# Notes



# Mass Dynamic: Problem Scenario



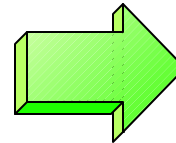
## Situation

- Using one HTTP server to host multiple domains
- Configuration file has many <VirtualHost> and inefficient

```
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>
```



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

# Notes



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 example in the presentation illustrates this situation:

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. Taking the advantages of the mass dynamic virtual host we are going to interpret the domain name. Based on the interpretation, the HTTP server will process the request. Using this new implementation, the HTTP configuration file looks like the figure on the left side of the presentation.

In the new configuration file, there is no ServerName directive, as this ServerName is provided by the URL received in the client request. The way the HTTP server identifies the ServerName provide in the header is based on the value configured to the UseCanonicalName directive as show in the table below.

UseCanonicalName value	Use
Off	The HTTP server will form a self-referential URL using the hostname 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 will use 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.

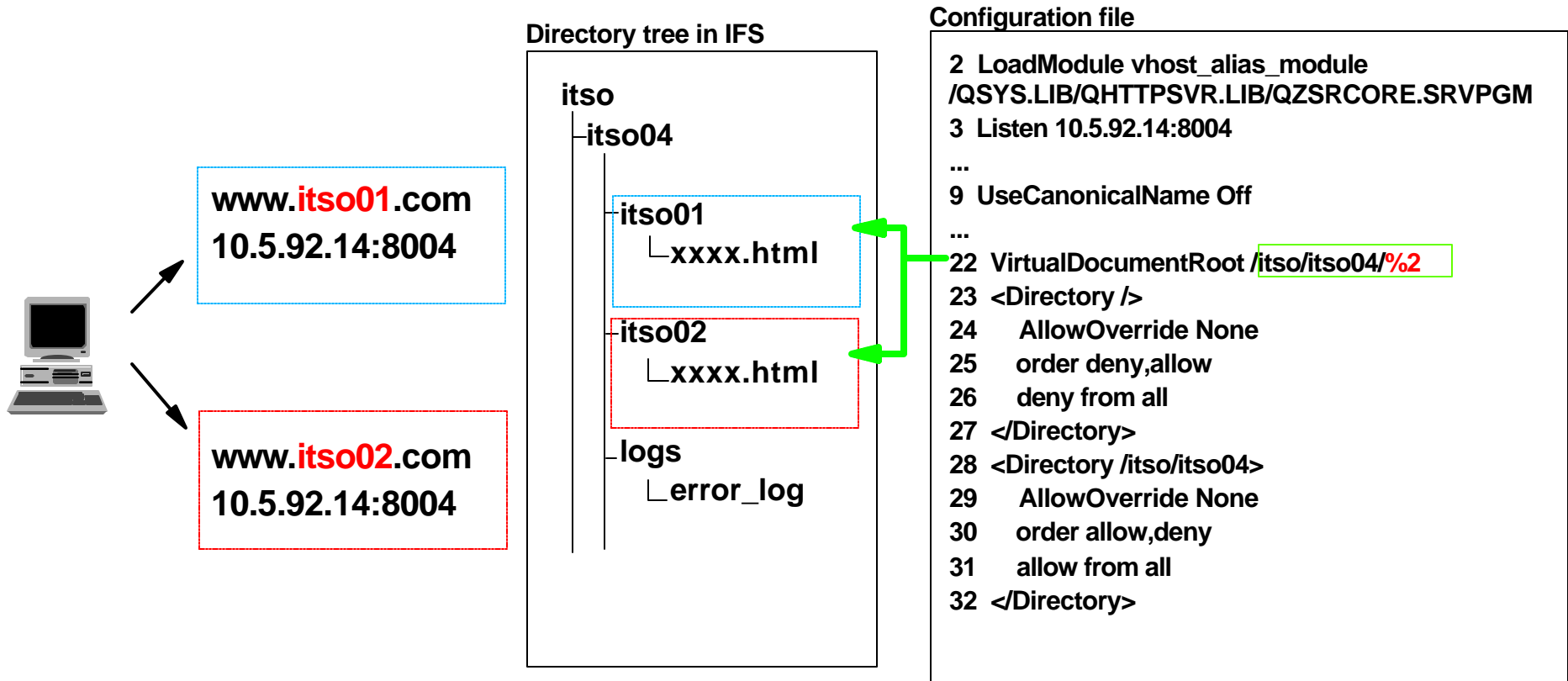
The advantages of the mass dynamic implementation are:

- Add domains dynamically.
- You do not need to restart the HTTP server in order to serve a new domain.

The disadvantages of this implementation:

- No individual logs when used with IP or named virtual host implementations.
- No tailoring of individuals domains with use of other directives in a virtual host context.

# Mass Dynamic: Solution



# Notes



To understand the advantages of the mass dynamic virtual host, we are going to act as a Internet Server Provider (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. What we need to do is 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

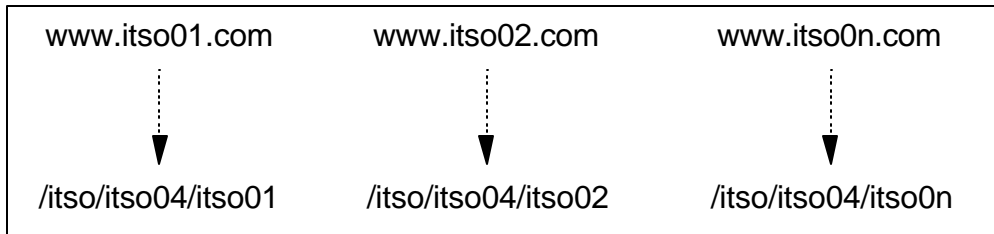
We need to find the appropriate interpolation value that allows to use the header provided in the URL, retrieve the host name and process the request. What we need to do is:

Retrieve the second part of the host name provided in the URL. In our case this will be the host name.

Interpolate the host name into some directory that exists in the iSeries IFS.

Serve the documents from that IFS directory.

In our example, the second part itsoXX, is part of the document root directive as shown in the figure below.



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

**VirtualDocumentRoot /itso/itso04/%2**

Where:

**/itso/itso04** is the document root of the HTTP server.

**%2** Retrieves the second part of the URL request. It is the directory used to process the requests. The place where the HTML code, images and so on are located.





# Security

# Notes



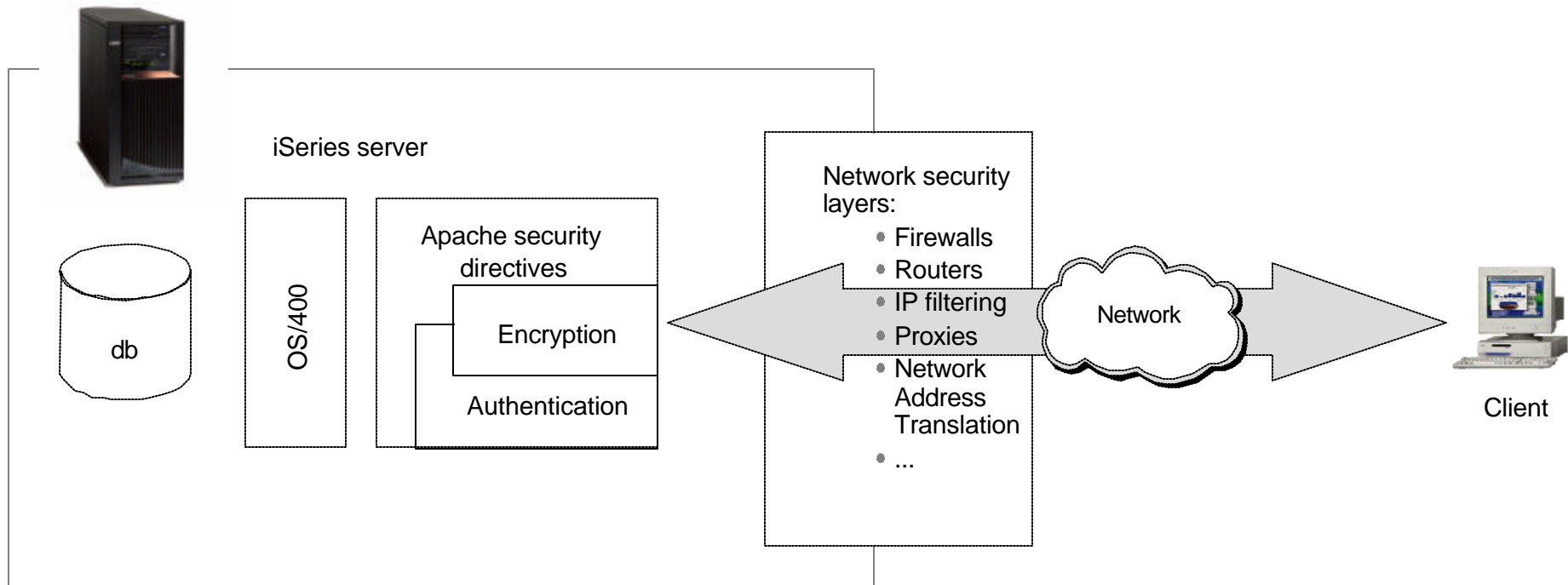
In this part, we will show you how to protect your server using user authentication, access control and encryption provided by the HTTP Server (powered by Apache) and the functions of OS/400.

- ▶ Security
- ▶ Basic authentication
- ▶ Basic authentication configuration
- ▶ Enabling SSL
- ▶ Client side digital certificates
- ▶ Proxy server

# Security



## iSeries security in the network environment



# Notes



Security is always one of the main concerns on the mind of a web server administrator. Even though your server will only run 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 making their way into intranets have repeatedly proved.

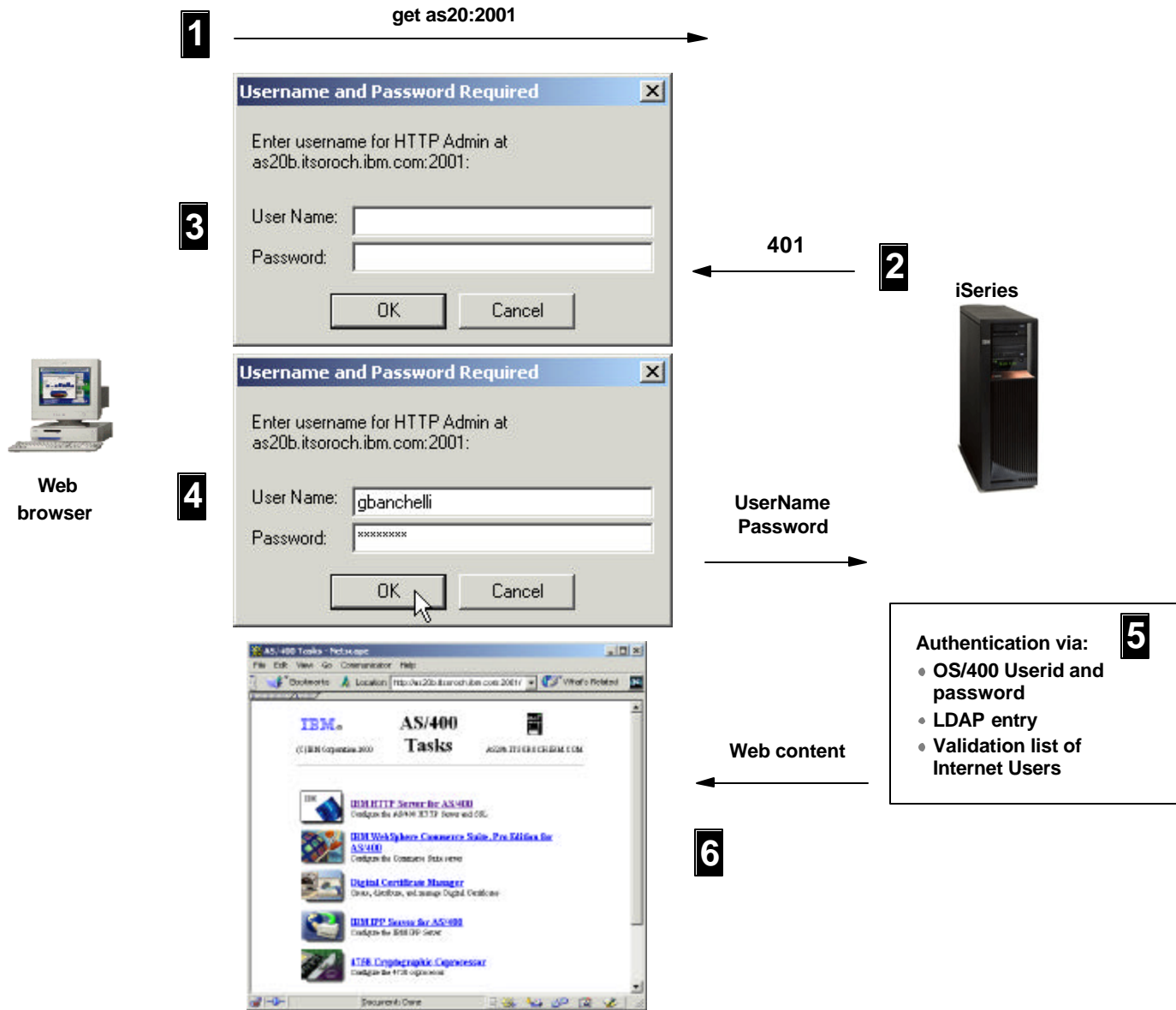
Security comes from a set of constantly updated rules and practices, specifically designed to protect the availability of your server and the integrity of your data. This figure presents a high level overview of iSeries server security in a network environment. A network security layer, encompassing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

Once inside, all requests are filtered by Apache server security: data is protected through user authentication, access control and encryption.

- Authentication is the process of verifying a user's identity through some sort of credentials. This can either be done through userid and password combinations or through an exchange of digital keys (or certificates).
- Access control (specifically, at this point we are discussing access control from the Apache HTTP server's point of view - above the access control that OS/400 also enforces) is enforced through a set of policies that define who can access your data, what kind of authority they will be granted, and what actions they will be allowed to perform on them. 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 is a mathematical process used to disguise data in order to keep unauthorized parties from gaining access to sensitive information. Data is encrypted into a ciphertext using a unique key and a set of operations that define an algorithm. Strength and effectiveness of encryption techniques depend on three factors: complexity of the algorithm, length of the encryption key, overall strength of the key itself. A compromised key could 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.

# Basic Authentication



Web browser



iSeries

# Notes



Basic authentication is a very popular means of verifying a user's identity before granting access to a protected resource, or realm. The figure above illustrates the authentication process, consisting of the following steps:

- 1 The client request access to a protected resource.
- 2 The server replies with HTTP status code 401 and a special header, WWW-Authenticate, containing 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.
- 5 Depending on the method you have chosen, those credentials are then checked against OS/400 user profiles, a validation list or 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.

# Basic Authentication Configuration



OS/400 user profiles

LDAP

Validation lists

```
34 <Directory /ITSO/itso06/itsoco/Projects/Archives>
35   AllowOverride None
36   AuthName MyRealm
37   Profile Token off
38   AuthType Basic
39   order allow,deny
40   allow from all
41   PasswdFile %SYSTEM%
42   require valid-user
43 </Directory>
```

# Notes



Although there are three different authentication methods, such as OS/400 user profiles, validation lists, and LDAP, these methods actually share more in common than have differences. This figure demonstrates this. This table reads top-down. By following your goals you can use the GUI configuration steps to create the Apache final configuration file directives as listed.

You must have the directives on lines 34, 36, 38 and 42. Your choices within these configuration directive lines are common no matter which 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 PasswdFile directive on line 41 is where you can make a choice as to a user access policy and a user validation policy.

**Authentication by OS/400 user profiles-** OS/400 user profiles can be used for authentication. The main advantage of this implementation is not requiring additional configuration steps or maintaining a separate user database. User profiles with limited capabilities and no signon access can be used for this purpose, as well as \*SECOFR class users (though this practice is highly discouraged).

**Tip:** Access validation through OS/400 user profiles is the simplest and least secure way of restricting access to your data. While acceptable in non-critical environments, this kind of authentication alone is not recommended on public networks such as the Internet, where its simple Base64 encoding and the use of actual user profiles and passwords could easily compromise the security of your system. A good choice for protecting your data would be SSL or TLS.

**Authentication by validation lists-** Protection by validation lists does not require use of actual OS/400 profiles and passwords, reducing risk to your iSeries server in the event a userid is compromised. As with all other forms of basic authentication, passwords are sent Base64 encoded. That is, sent in the clear.

**Authentication by LDAP entries-** The Lightweight Directory Access Protocol (LDAP) provides access to a centralized X.500 directory where information about users, networks and systems (actually any kind of information) is stored. Starting with V5R1M0 Directory Services (shipped as option 32 of the Operating System) are automatically installed with OS/400.



# Notes



In association with user authentication, you can specify a user profile that is used for giving an authorization to access the resources on iSeries.

This is specified by UserID directive. Though you can specify three types of values in this directive, the type of value you can specify depends on what value you have in PasswdFile directive.

The following table shows the relation between UserID and PasswdFile directives.

PasswdFile directive	UserID: %%SERVER%%	UserID: %%CLIENT%%	UserID: Specific userID of OS/400
%%SYSTEM%%	QTMHHTTP	Authenticated user	Specified user
File name of validation list	QTMHHTTP	n/a	Specified user*
%%LDAP%%	QTMHHTTP	n/a	Specified user*

Each data indicates the user ID that is used for resource authorization in iSeries.  
"n/a" means that combination isn't allowed.

When you specify %%SERVER%% in the UserID directive, you can use any of those three kind of password files. At that time, the user ID QTMHHTTP is used for authorization of the resources in iSeries.

When you specify %%CLIENT%% in the UserID directive, you can only use OS/400 user profile as the password file specifying %%SYSTEM%% in the PasswdFile directive. Because, in this case, the user ID authenticated by OS/400 security system is used for authorization of the resources in iSeries. This means that the user ID must be one of the OS/400 user profiles because any user IDs in LDAP or validation list cannot be recognized by OS/400 security system.

When you specify a specific user ID in the UserID directive, you can use any of those three kind of password files. In this case, the specified user ID must be one of the OS/400 user profiles so that it can be used for authorization of the resources in iSeries. Meanwhile, user authentication is executed using specified password file. This means that if you use LDAP or validation list, you must have the same user ID in your LDAP server or validation list as you specified in the UserID directive.

# Notes



# Enabling SSL/TLS



## HTTP server: ITSO06 global settings

### General Settings

Server IP address and port to listen on:

All	8006
All	44306

Virtual Host context: IP address \*:44306

### SSL General Settings

Enable SSL

Application name: QIBM\_HTTP\_SERVER\_ITSO06

add for enabling SSL

## CA managed by DCM

Certificate	Common name
SG24-6716	as20b.itsoroch.ibm.com

assign to digital certificate

# Notes



The figure above is an overview of the settings necessary for enabling SSL/TLS in your HTTP Server (powered by Apache). You can see the description of these settings in the table below. It lists a series of goals on the left. The first three are handled by the administration GUI of the HTTP Server (powered by Apache) and will result in new directives in your configuration file. The rest are dependant upon configuration steps necessary with the Digital Certificate Manager (DCM).

**Tip:** Since you need to add HTTPS as a new protocol for your server you must create a virtual host context in order to manage your SSL-secured communications.

Goal	GUI configuration steps	Apache final configuration file
Add a new port for SSL-secured communications	In your server's global settings panel select General Settings and add the new port number.	Listen 44306
Create a virtual host context that will contain the SSL directives	Through the Context Management menu 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 SSL General Settings and check the Enable SSL box.	2 LoadModule ibm_ssl_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM... 35 SSLAppName QIBM_HTTP_SERVER_ITSO06 36 SSLEnable 37 SSLCacheDisable 38 </VirtualHost>
Assign a digital certificate to the server	In the Digital Certificate Manager GUI open the *SYSTEM certificate store. Under Fast Path select Work with server applications and select your server from the list on the right. Assign a valid certificate and make sure that the CA is in 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 our virtual host can only be accessed through the HTTPS protocol now.	

# Notes



## Enabling SSL for the ADMIN instance

Enabling SSL support for the configuration GUI requires some additional considerations. First, you will have to add the following lines to the ADMIN customization include that is located in /QIBM/UserData/HTTPA/admin/conf/admin-cust.conf

```
LoadModule ibm_ssl_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM
Listen 2001
Listen 2010
SetEnv HTTPS_PORT 2010
<VirtualHost *:2010>
SSLEnable
SSLAppName QIBM_HTTP_SERVER_ADMIN
</VirtualHost>
```

Then issue the CL command:

```
CALL QHTTPSVR/QZHAPREG PARM('RegisterAppName' 'QIBM_HTTP_SERVER_ADMIN')
```

to register the ADMIN instance within the DCM environment.

You can now access the Digital Certificate Manager GUI and assign a server certificate to the QIBM\_HTTP\_SERVER\_ADMIN application you have just registered.

# Notes



# TLS Upgrade

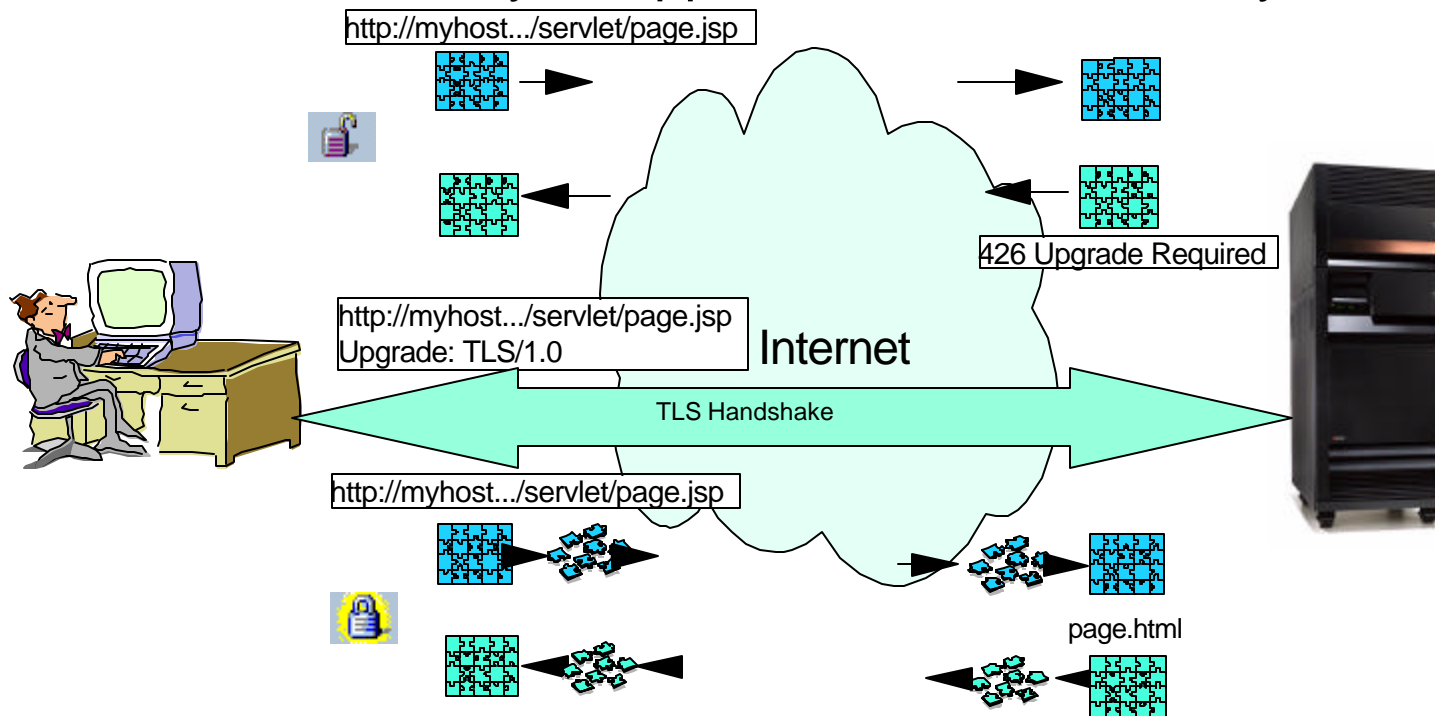


Allows client to request an upgrade to TLS encryption on an unencrypted port

- New applications only need 1 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:



# Notes



A new feature in 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 only need 1 port for both normal and SSL traffic. Right now the primary user is Internet Print Protocol (IPP).

Refer to **RFC 2817: Upgrading to TLS Within HTTP/1.1**. An excerpt:

The historical practice of deploying HTTP over SSL3 has distinguished the combination from HTTP alone by a unique 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 have similarly been requested -- and in some cases, granted -- to distinguish between secured and unsecured use of other application protocols (e.g. snews, ftps). This approach effectively halves the number of available well known ports.

At the Washington DC IETF meeting in December 1997, the Applications Area Directors and the 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 in order 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 will use the Host: header to disambiguate the intended web service. As HTTP/1.1 usage has grown more prevalent, more ISPs are offering name-based virtual hosting, thus delaying IP address space exhaustion.

TLS (and SSL) have been hobbled by the same limitation as earlier versions of HTTP: the initial handshake does not specify the intended hostname, 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 -- will allow ISPs to provide secure name-based virtual hosting as well.



# Client Side Digital Certificates



## HTTP server: ITSO06 global settings

Virtual Host context: IP address \*:44306

### SSL Client Authentication

Require valid certificate for connection

### Directory /ITSO/itso06/itsoco/Projects/Project\_3

#### Basic Authentication

User name to process requests: %%CLIENT%%

User authentication method to validate passwords: Use user profile

#### SSL Client Authentication

Use SSL client authentication

Require valid client certificate for accessing this resource



Client certificate



# Notes



Client side digital certificate is 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.

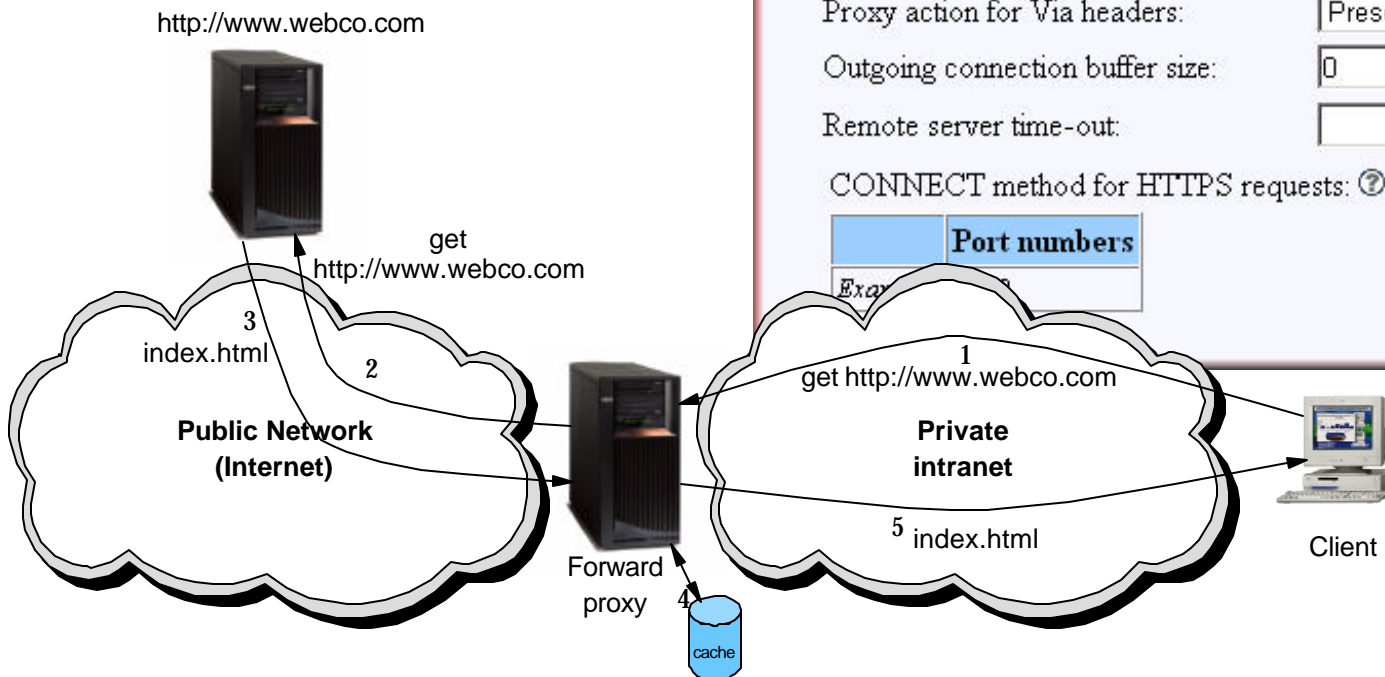
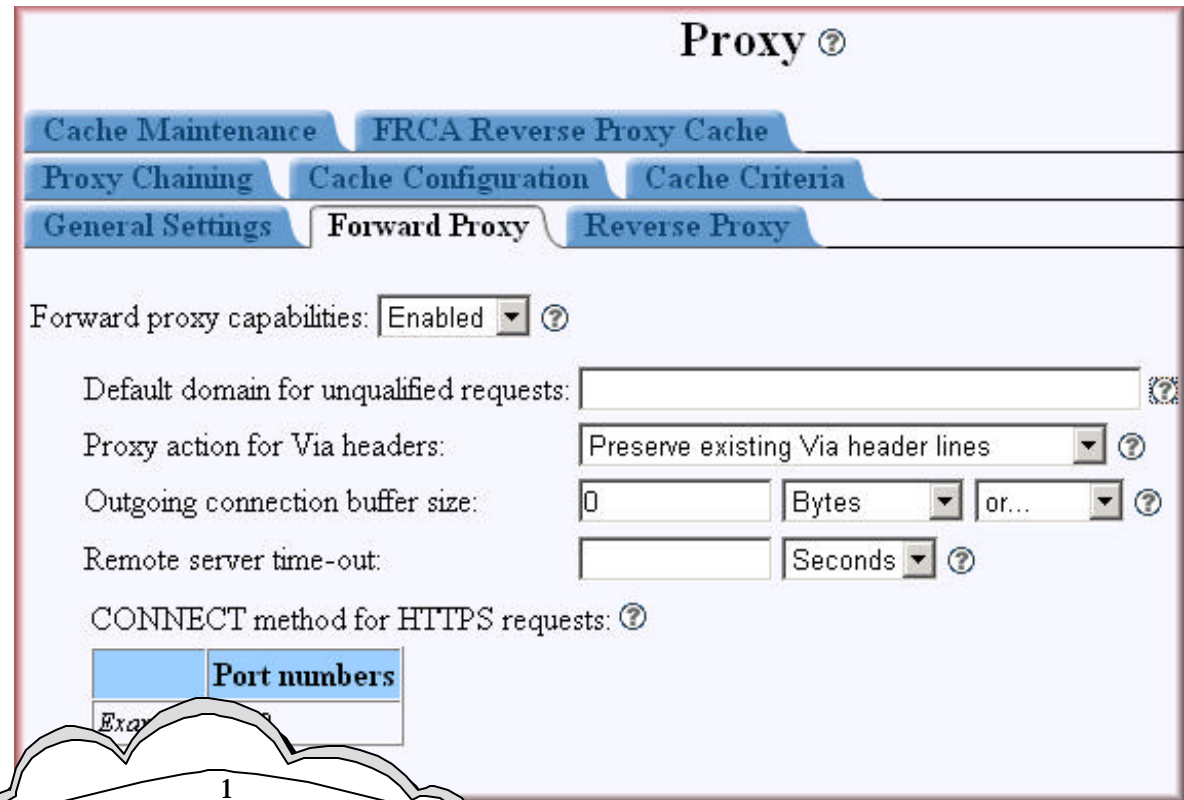
This figure is an overview of definition example. The folder we want to protect will be served by a Directory directive inside our virtual host context.

This configuration requires a valid client certificate for access to the new folder, but also forces the server to access protected data as the user for whom that certificate has been issued. This powerful capability of the Apache server, also known as profile swapping, is yet another proof of the granularity and versatility of Apache security.

# Proxy Server



- Forward (shown)
- Reverse
  - FRCA
- Proxy Chaining



# Notes



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 as a single point of access for communications with un-trusted networks. Proxies can also dramatically improve HTTP response times by serving documents from a local cache, effectively cutting down network traffic, bandwidth occupation and CPU load (depending on the type of request being served).

There are two mainstream proxy implementations: the forward proxy and the reverse proxy. Both can be implemented as virtual hosts or standalone servers. Apache proxy can also be configured as part of a proxy chain by specifying which server the requests will be relayed to.

**Forward Proxy-** A forward proxy fetches content from another server, allowing clients to reach a network they wouldn't otherwise have access to. Above figure demonstrates the role of a forward proxy in an environment where clients on a private intranet do not have direct access to the Internet.

In this configuration the clients send all outbound HTTP requests to the forward proxy ( ). The proxy checks the request against security restrictions, 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 ( ) and retrieves an updated copy of the document ( ). The document is (optionally) stored in the local cache ( ) and sent to the requestor ( ). 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.

# Notes



**Reverse proxy-** Two other forms of proxy support are available with the HTTP Server (powered by Apache). One is reverse proxy which is the same as a forward proxy except that requests from outside of the firewall to the proxy are allowed. The other is proxy chaining which requires two or more proxy servers and can be used to balance server workloads or network traffic.

Another reverse proxy is available with your HTTP Server (powered by Apache) and that is FRCA. FRCA will be covered later in this presentation.

A reverse proxy is another common form of a proxy server and 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 lessen 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. One 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 will first check to make sure a request is valid. If a request is not valid, or not allowed (blocked by the proxy), it will 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 will request the information from the content server and serve it to the requesting client. It also caches the information for future requests.

# Notes



**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 would not be possible or permitted. For example, a request using HTTP is sent to a server that can only handle FTP requests. In order for the request to be processed, it must pass through a server that can handle both protocols. This can be accomplished by making use of proxy chaining which allows the request to be passed from a server that is not able to fulfill such a request (perhaps due to security or networking issues, or its own limited capabilities) to a server that can fulfill such a request.

The first proxy server in a chain will check to make sure a request is valid. If a request is not valid, or not allowed (blocked by the proxy), it will reject the request resulting in the client receiving an error or a redirect. 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 will pass the request on to the next proxy server in the chain. This server also has the ability to fulfill, forward, redirect, or reject the request. If it acts to forward the request then it too passes the request on to yet 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 whatever protocol is required, to obtain the information. The information is then 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, proxy chaining may be used to get information to pass through multiple networks where a client on one network cannot communicate directly with a proxy server on a different network and needs a second proxy to relay its requests. It may also be used to cause information to be cached in multiple locations or to allow certain protocols to be used outside a firewall which cannot be allowed through a firewall.

For more information about proxy support by the HTTP Server (powered by Apache) see the Documentation Center. Go to <http://www.ibm.com/eserver/series/software/http> and then select Documentation.



# Serving Dynamic Data

# Notes



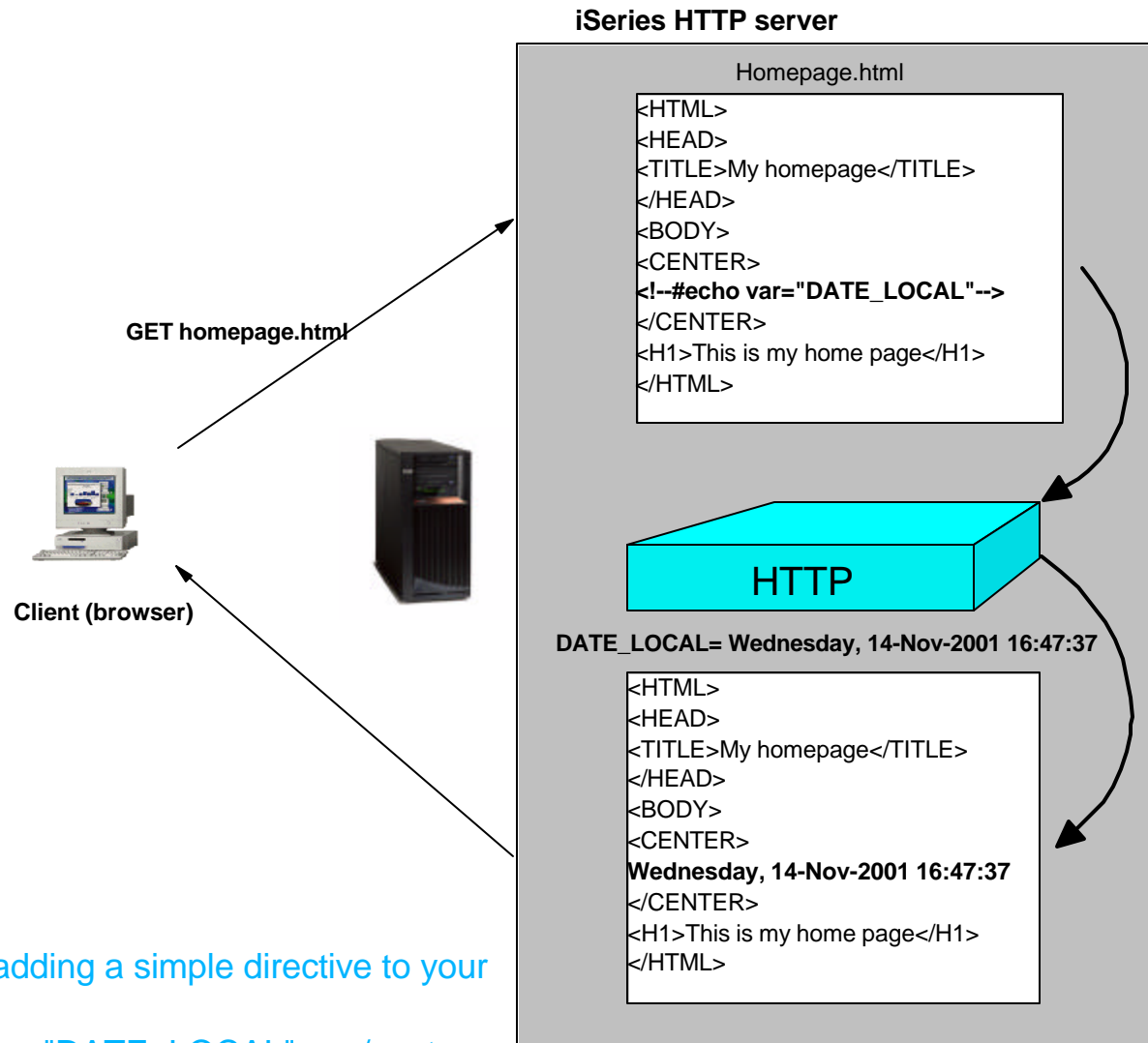
The IBM HTTP Server (powered by Apache) supports the most popular techniques generally available for this purpose such as Server-Side Includes (SSI), Net.Data, Common Gateway Interface (CGI).

In this part , we will briefly describe how to implement these functions the IBM HTTP Server (powered by Apache) .

- ▶ Server Side Includes (SSI)
- ▶ Net.Data
- ▶ CGI



# Server Side Includes (SSI)



You can test SSI by adding a simple directive to your HTML file like:  
<center><!--#echo var="DATE\_LOCAL"--></center>

# Notes



Server-side includes (SSI) are the simplest way of adding dynamic content to a web site. A set of directives are embedded in the HTML code and are interpreted by the server before the document is sent to a client. SSI can be used to trigger a CGI program, return information about documents, or the value of environment variables, as shown in the figure.

In a simple sense SSI allows for character substitution from within an HTML document.

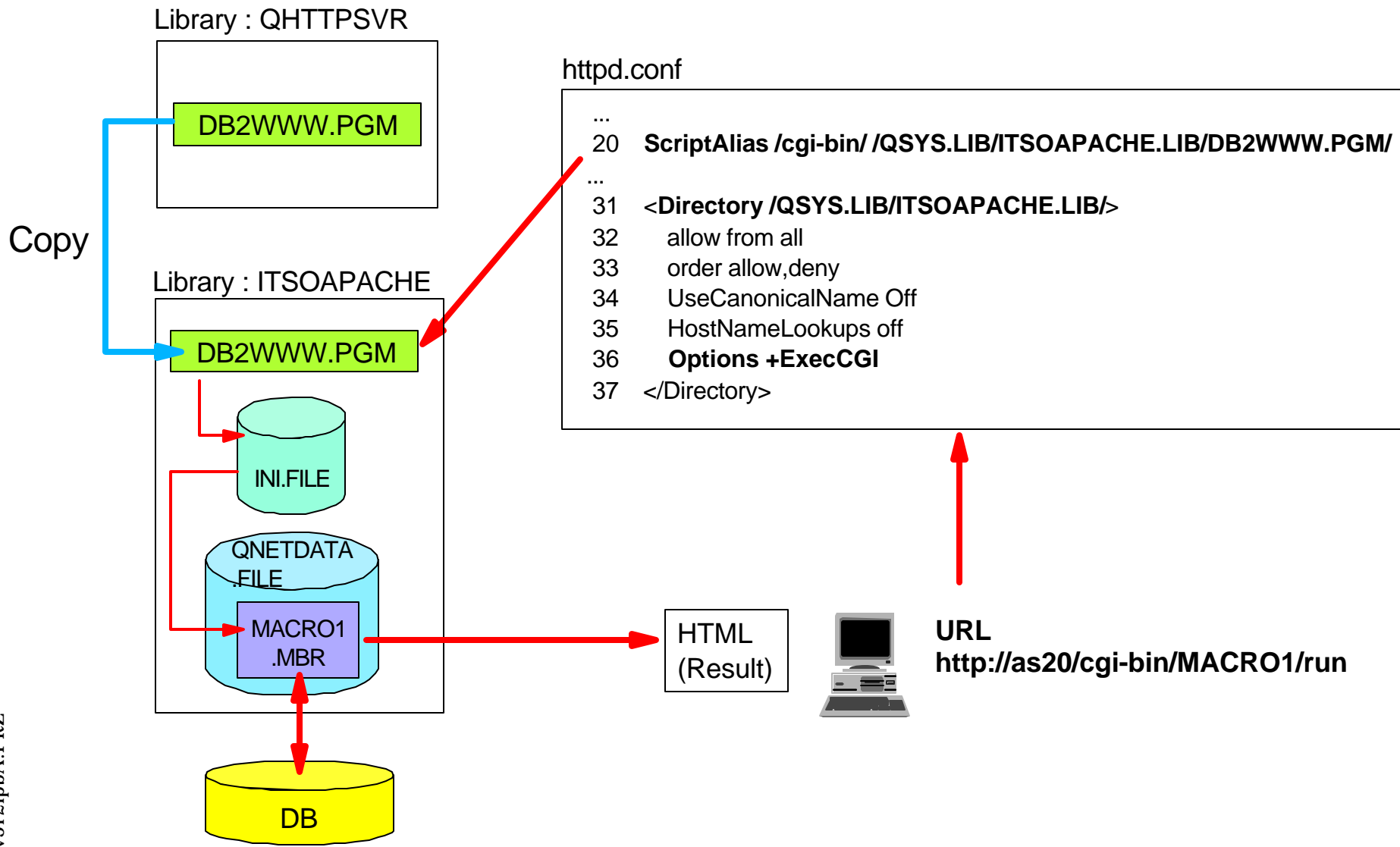
SSI also supports the execution of simple conditional statements, thus providing a reasonably flexible programming environment. The syntax used for SSI directives is `<--#command parameter="value" -->`. This syntax allows SSI directives to remain hidden when the server is not configured for SSI support.

**Tip:** Special characters inside SSI directives will have to be preceded by a backslash (\).

This table lists the SSI commands available on iSeries and their respective parameters. Additional information is provided in the Reference section of the HTTP Documentation Center that can be found at [http://publib.boulder.ibm.com/pubs/html/iseries\\_http/v5r1/index.htm](http://publib.boulder.ibm.com/pubs/html/iseries_http/v5r1/index.htm)

Command	Description	Valid parameters
config	Controls various output formats.	errmsg, sizefmt, timefmt
echo	Prints one of the SSI or API variables. Dates are printed using config timefmt.	var, encoding
exec	Calls a CGI program.  Note: shell command execution is not allowed in the OS/400 environment.	cgi
fsize	Prints the size of the specified file according to config sizefmt.	file, virtual
flastmod	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

# Net.Data



# Notes



Net.Data is an easy to use scripting language developed by IBM and bundled with the IBM HTTP server . Net.Data macros are fed to a CGI interpreter (DB2WWW) that generates HTML output. The Net.Data macro language allows you to generate a wide range of dynamic content through imbedded dynamic SQL invocations and program calls.

A Net.Data environment is made of the macro processor, a configuration file (known as initialization or INI file) and the user code.

The first thing you will do is create a user library to have a place to keep these files. In this figure of example, we created library ITSOAPACHE using the Create Library (CRTLIB) command.

Next, you should copy the original macro processor program DB2WWW from library QHTTSPVR to your library using the Create Duplicate Object (CRTDUPOBJ) CL command. 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.

The next step will be the creation of the Net.Data initialization file, which will reside in the same library as the macro processor. Use the Create Source Physical File (CRTSRCPF) CL command to create file INI with a DB2WWW member. Note that since all the statements in the INI file will have to be on single lines, 240 is the recommended record length value.

The initialization file will contain server your default environment settings like the path where macros are stored, environment variables, logging and tracing preferences, and much more. Use Start PDM (STRPDM) to modify the INI file.

See the Net.Data Administration and Programming Guide for OS/400 and Net.Data Reference manuals at <http://www-1.ibm.com/servers/eserver/series/software/netdata/docs/doc.htm> for additional information on environment settings.

Macros can be stored on the iSeries as members of a source physical file in a library or as stream files (usually with .d2w or .ndm extension) inside the IFS. The two solutions are equivalent. Choose one and change the MACRO\_PATH statement in your configuration file to reflect your choice.

To make the HTTP server (powered by Apache) available for Net.Data, you need to define ScriptAlias directive and directory context for the library which has DB2WWW program and INI file. In this directory context, you have to have `Options +ExecCGI` for enabling CGI.



## The HTTP Server (powered by Apache) provides CGI support for:

- C/C++, (Perl), Java, REXX, RPG and COBOL
  - Note: Perl scripts will work but Perl is not supported by IBM
- CGI programs in PASE
- Persistent Connections
  - allows multiple objects to be served over a single connection
  - reduces overhead of starting new connections
  - allows complete transactions to be done with a single connection
- Prestart and reuse jobs for CGI
- Multithread capable CGI support
- CGI Environment Variables
  - JavaClassPath
  - LibraryList

# Notes



Common Gateway Interface (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 iSeries HTTP server can be written in C++, Rexx, Java, ILE C, ILE RPG, or ILE COBOL.

The iSeries can even run a CGI application that has been written and compiled for AIX. The binary output of the compiler is moved to the iSeries and can run from the Portable Application Solution Environment (PASE).

CGIs come into play whenever a significant processing load has to be employed to generate dynamic output.

Persistent connections allow multiple sequential HTTP requests to be made by a client over the same connection if the client indicates that it is capable of doing so. If a page contains many parts, such as individual graphics, this allows them to be sent over a single connection instead of incurring the overhead of opening multiple connections. This also allows a much more rapid dialog between the server and the client in CGI applications by maintaining active database transactions and persistent data across multiple HTTP requests.

In generic Apache implementations, a persistent connection ties up the server process until the connection is ended, usually through the timeout set with the *KeepAliveTimeout* directive. For iSeries, a new asynchronous I/O model allows support for persistent connections without tying up the server threads. This should provide a much faster response in busy servers.

Persistent connections are set with the *KeepAlive* directive, which defaults to *On*. Most Internet browsers support persistent connections, but some older browsers do not. The configuration shipped with generic Apache, and those created with iSeries' wizards contain *Browsermatch* directives that cause the server to revert to non-persistent HTTP 1.0 connections when one of the problem browsers is detected.

The *StartCGI* and *StartThreadedCGI* directives can be used to designate an initial number of CGI jobs to start at server startup time. Prestarting these jobs will eliminate the overhead of creating new ones as required.

New server functions allow setting of the *JavaClassPath* and the *Library List* for CGI jobs. This will allow users to easily configure test environments by allowing easier substitution of test libraries and Java Classes.

The best implementation guide for CGI programming both with the HTTP Server (original) and HTTP Server (powered by Apache) can be found in the HTTP Server for iSeries Web Programming Guide (Version 5 December, 2001 - GC41-5435) found at <http://www.ibm.com/eserver/iseries/software/http/docs/doc.htm>

Information about the unsupported version of Perl for the iSeries can be found in a Technical Studio article <http://www.iseries.ibm.com/tstudio/workshop/tiptools/perl.htm> for a freeware version of Perl that has been ported to the iSeries.

# PHP scripts as a CGI



## Hypertext Preprocessor (PHP)

- Powerful scripting language (think: "If Net.Data was open source")
- Apache based
- Two runtime engines:
  - As an Apache module
  - As a CGI



## Redpaper: Bring PHP to Your iSeries Server, REDP3639

- published January-29-2003
- Demonstrates: download, compile, configuration and test of PHP script
- OS/400 V5R2 (V5R1 OK too)
- PHP runtime engine is PHP as a CGI running in PASE

# Notes



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 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.

To “run” PHP scripts with your HTTP Server (powered by Apache), a PHP engine is required on your IBM iSeries server. The PHP engine is an open source product, so this IBM Redpaper demonstrates the process 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-BIN. Support for PHP as a module is not yet available for OS/400. The step-by-step implementation discussed in this redpaper involves the CGI version of PHP running in OS/400's Portable Application Solutions Environment (PASE). 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.

If you want to advertise this Redbook to your Customers and other Business Partners send them too:

- IBM intranet: <http://w3.itso.ibm.com/abstracts/redp3639.html>
- Internet: <http://www.redbooks.ibm.com/abstracts/redp3639.html>





# Web Application Serving

# Notes

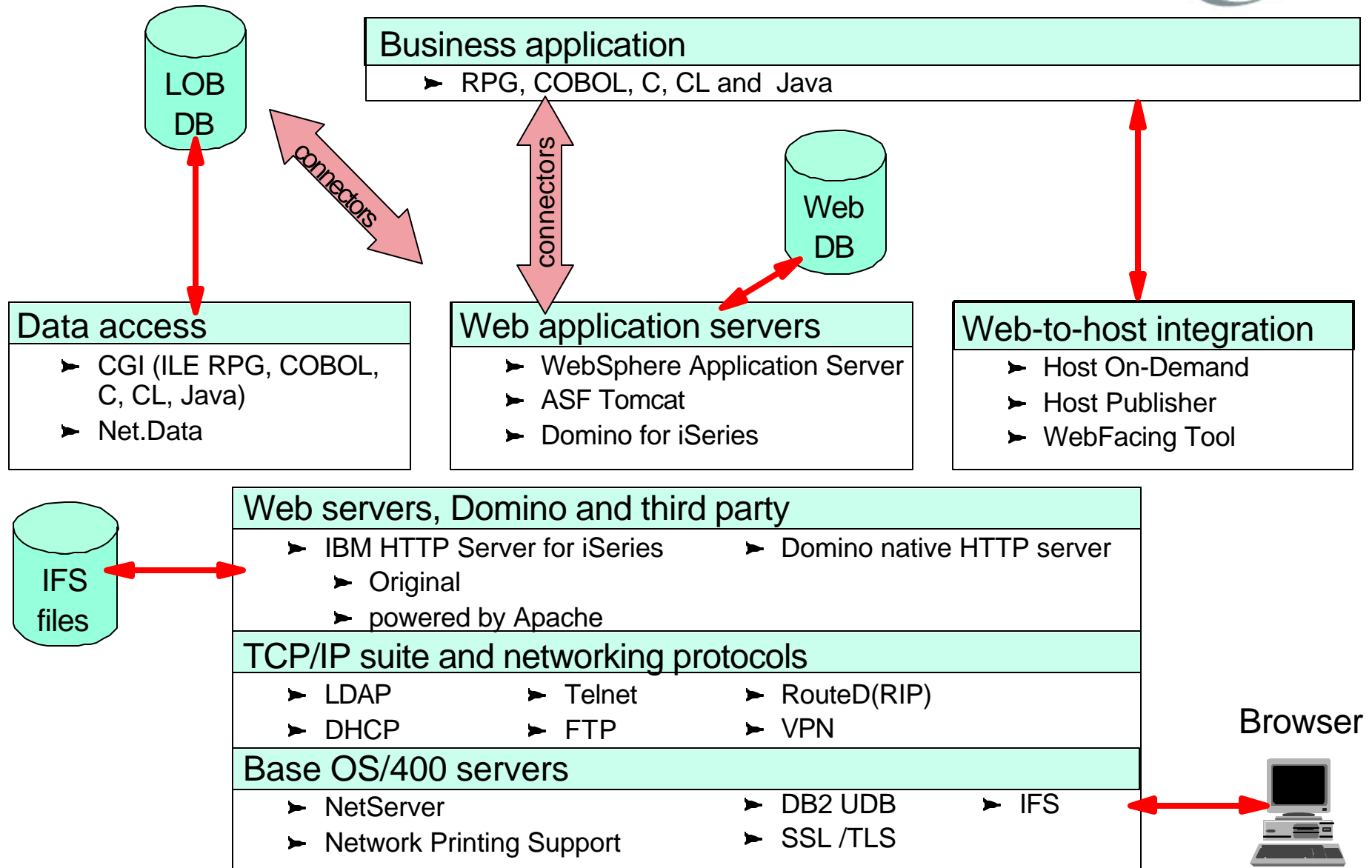


The HTTP Server (powered by Apache) includes an industry-standard Java servlet and Java Server Pages (JSP) engine based on technology from the Apache Software Foundation's Jakarta Tomcat (ASF Tomcat) open source code base. Lightweight and easy-to-use software extends the HTTP Server (powered by Apache).

This part shows you an outline of ASF Tomcat and also shows you some configuration related information.

- ▶ Web application serving
- ▶ ASF Tomcat on iSeries
- ▶ When to use ASF Tomcat
- ▶ ASF Tomcat directory structure
- ▶ ASF Tomcat directives
- ▶ ASF Tomcat authorities
- ▶ ASF Tomcat log files

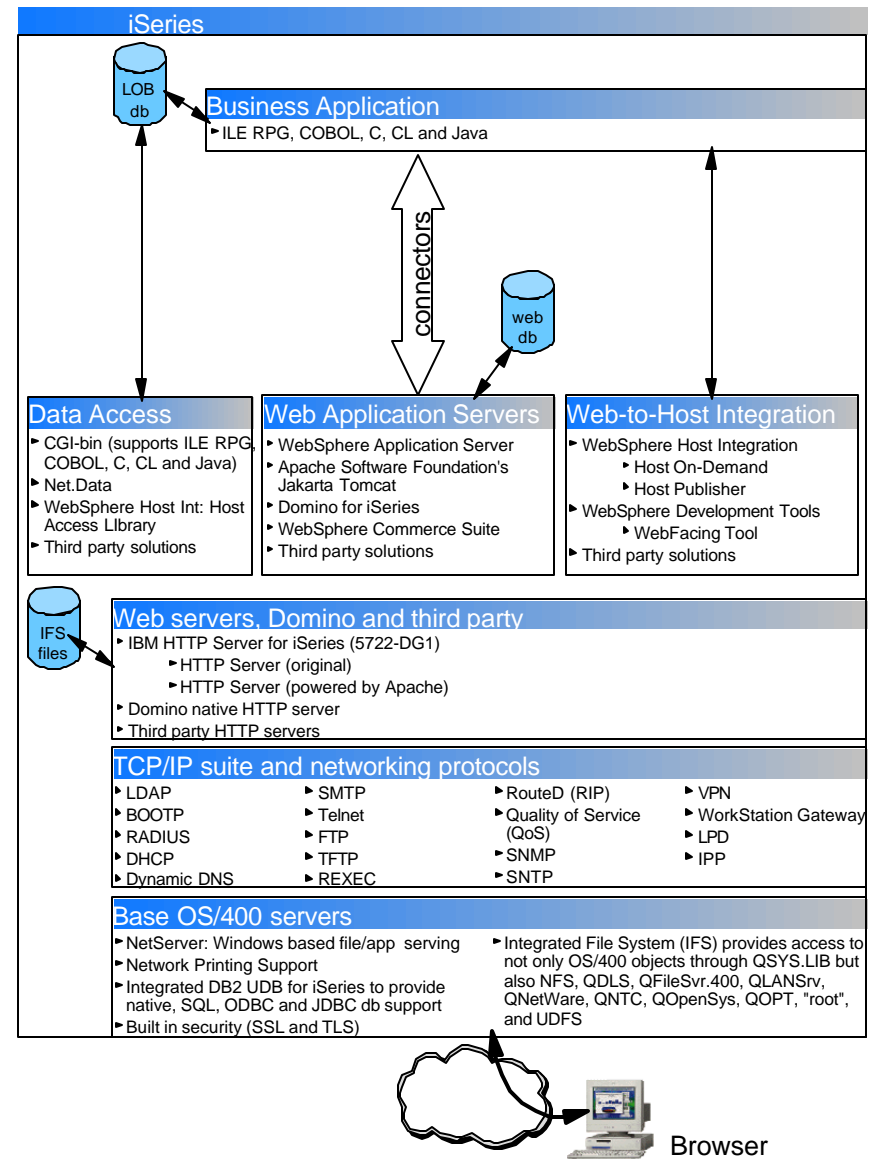
# Web Application Serving



# Notes



As you can see, there are many options to create, serve and manage one e-business application using the iSeries system. The iSeries aggressively supports the transformation of business applications to an e-business model, while also minimizing disruption within the enterprise environment. It has business-proven values (reliability, security, scalability, low cost of ownership) that supports the latest enabling technologies for e-business. In combination, these two qualities make iSeries an excellent choice for not only extending existing applications but also for deployment of new solutions.



# ASF Tomcat on iSeries



## Apache Software Foundation Jakarta Tomcat (ASF Tomcat)

- Servlet engine container
- Supports servlets, JSP, WAR files
- Integrated in IBM HTTP Server for iSeries (5722-DG1) base code

## Run in two process modes

- In-process
  - Run in the same process of the HTTP Server
  - Communicate through Java Native Interface
- Out-of-process
  - Run in separate process from the HTTP Server
  - Communicate through TCP/IP

# Notes



Apache Software Foundation Jakarta Tomcat (ASF Tomcat) is a servlet engine container that supports servlets, JavaServer Pages (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 system by the IBM HTTP Server for iSeries base code and not by a new option or License Program Product. ASF Tomcat requires a Java Runtime Environment that has conformity with JRE 1.1 or later, including any Java2 platform system.

The iSeries server supports ASF Tomcat version 3.2.4 on V5R1 and 3.2.1 on V4R5.

For ASF Tomcat to work with the HTTP Server (powered by Apache) needs an “agent” that resides in the HTTP server and send him servlet request. This “agent” is the Web server plug-in, `jk_module`. This module allows the communication between the HTTP Server (powered by Apache) and the ASF Tomcat servlet engine and 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 the HTTP Server (powered by Apache), this does not means that the servlet engine needs to run in the same process as the HTTP server. ASF Tomcat can be configured to run:

**In-process** ASF Tomcat and the HTTP Server (powered by Apache) run in the same process and communicate through a Java Native Interface (JNI).

**Out-of-process** ASF Tomcat and the HTTP Server (powered by Apache) run in separate process (even on separate systems) and communicate through TCP/IP sockets. The ASF Tomcat server process runs in the QSYSWRK subsystem.

# Notes



Running in-process or out-of-process implies some differences, as shown in this table.

In-process	Out-of-process
Uses the <code>jk_module</code> module with Java invocation API to communicate with the HTTP server	Uses the <code>jk_module</code> to communicate with the HTTP server.
The HTTP server and ASF Tomcat servlet engine communicate through a JNI	The HTTP server and ASF Tomcat communicate through TCP/IP sockets
Does not require a new protocol	Does require a new protocol to communicate ( <code>ajp12</code> and <code>ajp13</code> )
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 configure by the HTTP server	Uses his own container managed security implementation
Works with the SSL configuration of the HTTP server	The communication between the HTTP server and ASF Tomcat does not support SSL

# Notes





# When to Use ASF Tomcat



## When the customer's application uses

- Servlets, JSP and XML files

## and does not require

- EJB support
- DB connection management
- SSL or other security mechanism between the HTTP server and the application server
- Load balancing implementation
- Scalability

# Notes



Some iSeries customers want a basic, no-cost Web application server that supports Servlets and JavaServer Pages. Relying on IBM's HTTP Server (powered by Apache) as its web server, the Apache Software Foundation's Jakarta Tomcat provides a basic web application server for iSeries customers. You can use ASF Tomcat:

- When your application is based on servlets, JSP and XML files.
- When your application does not require EJB support.
- When your application does not require any database connection manager mechanism.
- When your application does not require any specific security mechanism, for example SSL, between the HTTP server and the application server. (This restriction is only applied to the "out-of-process" mode.)
- When your e-business solution does not require a load balancing implementation.
- When your e-business solution does not require scalability.

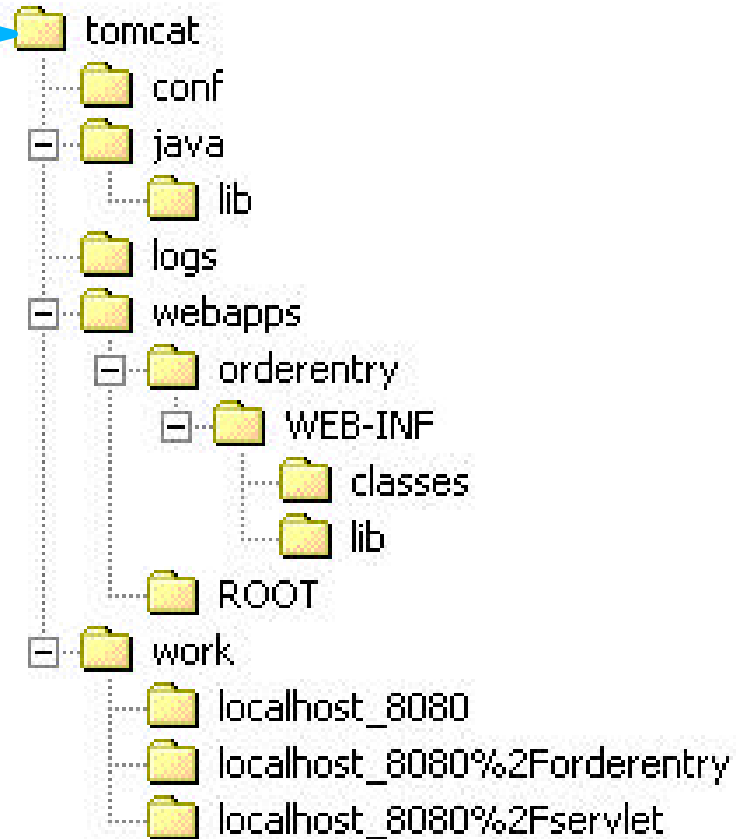
ASF Tomcat is the newest component in the e-business solution provide by the iSeries server. In the following section we will learn more about this new member. If you want to learn more about WebSphere Application Server, refer to:  
<http://www.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/>

# ASF Tomcat Directory Structure



Located in root or QOpenSys in IFS

Home directory



# Notes



This servlet engine, runs under his own directory structure. This directory structure is used by the HTTP server as it is include into the HTTP configuration file. The directory structure can be located in the root or QOpenSys file systems. The following table shows you the directory structure used by ASF Tomcat.

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 director
tomcat_home/webapps	This directory contains WAR files if you have them. All WAR files are expanded and subdirectories are added as contexts
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 document base directory. 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 dire
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.

# Notes



ASF Tomcat directory	Description
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.

# Notes



# ASF Tomcat Directives



## ASF Tomcat and HTTP configuration file (out-of-process)

```
HTTP server:  ITS008
...
2  LoadModule jk_module /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM
3  Listen 10.5.92.14:8008
4  DocumentRoot /itso/itso08/itsoco
...
21 JKMount /orderentry/* remote
22 JkAsfTomcat On
23 JkWorkersFile /itso/itso08/conf/workers.properties
24 JkLogFile /itso/itso08/logs/jk.log
25 JkLogLevel error
...
```

workers.properties file

```
worker.list=remote

worker.remote.type=ajp13
worker.remote.host=10.5.92.14
worker.remote.port=8009
```

Servlet engine

# Notes



## ASF Tomcat directives

The ASF 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 Tomcat to be turned off without deleting particular ASF Tomcat directives from the HTTP Server (powered by Apache) configuration file. When this directive is set to Off, it appears to the user as if ASF 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 what detail of logging should occur to the log file defined by `JkLogFile`. This possible values for this directive are:

- debug
- info
- error
- emerg

**JkMount-** The `JkMount` directive specifies which URI contexts are sent to a ASF 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 Tomcat will not function. The typical file name is `workers.properties`.

These directives are added into the HTTP configuration file when the ASF Tomcat configuration is created. The directives may look like the following:

```
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
```



# Notes



## The workers.properties file

The worker is the ASF Tomcat instance that runs to serve servlets and JSP request coming from the web server, in our case, coming from the HTTP Server (powered by Apache). This worker can run In-process or out-of process. This worker is specified in the JkWorkersFile directive and tells to the HTTP Server (powered by Apache) how the ASF Tomcat instance runs. This file contains entries of the following form:

```
worker.list=<a comma or space separated list of worker name>, for example:  
worker.list=local, remote  
or  
worker.list=local remote
```

When starting up, the web server plug-in (jk\_module) will instantiate the workers whose names appears 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 Tomcat process. The available workers types are:

- ajp12** This worker knows how to forward request to out-of-process ASF Tomcat process using the ajp12 protocol.
- ajp13** This worker knows how to forward request to the out-of-process ASF Tomcat process using the ajp13 protocol.
- jni** This worker knows how to forward request to the in-process ASF Tomcat process using Java Native Interface (JNI).

The differences between the ajp12 and ajp13 protocols are:

- The ajp13 protocol is a binary protocol and it will attempt to compress some of the requested data.
- The ajp13 protocol reuse open sockets and leaves them open for future requests.
- The ajp13 protocol has special treatment for SSL information.

Defining workers of certain type should be done with the following property format:

```
worker.<worker name>.type=<worker type>, for example  
worker.local.type=jni, for ASF Tomcat in-process mode, where local is the name of this worker.
```

# Notes



Each one of these workers has its own group of properties which define the ASF Tomcat attributes like: ports, hostnames, classpaths, and so on. The attributes are different if running in-process or out-of-process and if using ajp12 or ajp13 types. Depending of your ASF Tomcat run mode, the workers.properties file should have entries like the following:

```
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/HTTPPA/java/lib/webserver.jar
```

This workers.property file specify: in-process mode, with Java 1.2 and Tomcat home directory /itso/itso07.

```
worker.list=remote
worker.remote.type=ajp13
worker.remote.port=8009
worker.remote.host=localhost
```

This workers.property file specify: out-of-process using ajp13 protocol, on port 8009 and running in the local in the iSeries server.

When you create the ASF Tomcat servlet engine using the ASF Tomcat wizard provide with the HTTP Server (powered by Apache) server, those entries are created in the workers.properties file for you. So, at this point this information is only supply as a reference.

If you want to learn more about the workers.property file and his properties refer to:

- Tomcat Workers How To at the Apache Software Foundation's Web site:  
<http://jakarta.apache.org/tomcat/tomcat-3.2-doc/Tomcat-Workers-HowTo.html>

# Notes



# Notes



A fantastic example of the power of the open source community is the newly released module jk. jk is a replacement to the elderly mod\_jserv. It was a completely new Tomcat-Apache plug-in that handles the communication between Tomcat and Apache.

The newest jk2 is a rewrite of jk. The native part has been completely restructured and the configuration has been simplified a lot.

Why should you use the jk?

- jk was developed to overcome many limitations of its ancestor, mod\_jserv.
- mod\_jserv was too complex and because it was ported from Apache/JServ, it brought with it lots of JServ specific bits that aren't needed by Apache.
- Where mod\_jserv supported only Apache web servers on Unix OS, jk supports much more web servers and operating systems through via a compatibility layer named the jk library. The layered approach provided by the jk library makes it easier to support many different web servers and OS.
- jk offer better support for SSL, that's was a problem with mod\_jserv which couldn't reliably identify whether a request was made via HTTP or HTTPS.
- jk can, using the newer Ajpv13 protocol which relay many SSL informations required by servlet 2.2 and 2.3 specs.
- jk offers a lot of different and flexible communications between a Web Server and the Tomcat Servlet Engine and could be used today with all of the ASF Tomcat Engines, 3.2.x , 3.3.x , 4.0.x , 4.1.x and 5.x

Supported Configuration: The mod\_jk module was developed and tested on:

- Linux, FreeBSD, AIX, HP-UX, MacOS X, and should works on major Unixes platforms supporting Apache 1.3 and/or 2.0
- WinNT4.0-i386 SP4/SP5/SP6a (should be able to work with other service packs), Win2K and WinXP and Win98
- Cygwin (until you have an apache server and autoconf/automake support tools)
- Netware
- **iSeries V5R1 and V5R2 with Apache 2.0.39. Be sure to have the latest Apache PTF installed.**
- Tomcat 3.2.x, Tomcat 3.3.x, Tomcat 4.0.x, Tomcat 4.1.x and Tomcat 5

For more information and directions on where to download the source and how to compile and configure the jk see It will be included in Tomcat 4.1 official documentation: <http://jakarta.apache.org/tomcat/tomcat-4.1-doc/jk2/index.html> (as of 9/9/2002 the documentation is not there yet - should be by the end of September).

# ASF Tomcat Authorities



## Considerations for out-of-process

### The user profile to configure Tomcat must have

- \*JOBCTL authority
- \*ALL authority to the file QUSRSYS/QATMHASFT
- \*CHANGE authority to the library object QUSRSYS

### The user profile to start Tomcat must have

- \*USE authority to the file QUSRSYS/QATMHASFT
- \*USE authority to the profile associated with the server user profile
- \*IOSYSCFG special authority

### The user profile Tomcat runs under must have

- \*USE authority to the file QUSRSYS/QATMHASFT
- Must not have
  - \*SECADM authority
  - \*ALLOBJ authority

# Notes



Out-of-process ASF Tomcat configurations have the following authority considerations:

The user profile configuring the out-of-process ASF Tomcat is the owner of the configuration files that are created. This user profile must have:

- \*JOBCTL authority
- \*ALL authority to the file QUSRSYS/QATMHASFT
- \*CHANGE authority to the library object QUSRSYS

In addition the configuration process creates the tomcat\_home directory with public execute authority. The default out-of-process tomcat\_home directory is /ASFTomcat/tomcat\_server\_name. If any of these directories existed prior to the ASF Tomcat configuration process, then the previous authorities are left unchanged.

The user profile used to start the out-of-process ASF Tomcat must have:

- \*USE authority to the file QUSRSYS/QATMHASFT
- \*USE authority to the profile associated with the server user profile (this is QTMHHTTP by default).
- \*IOSYSCFG special authority

By default the user profile that the out-of-process ASF Tomcat runs under is the QTMHHTTP user profile, but you can configure this to be another user profile.

- This user profile must have \*USE authority to the file QUSRSYS/QATMHASFT.
- This user profile must NOT have:
  - \*SECADM authority
  - \*ALLOBJ authority (if the system is at security level 30 or greater).

In-process ASF Tomcat configurations have the following authority considerations:

When running JSPs on an in-process ASF Tomcat, in order to assure that the .java and .class files resulting from the compilation process of a JSP are owned by the configured profile for that server, the JSPs should be precompiled by the server administrator under that configured profile. This will assure users swapped to by the HTTP Server are not the first to cause the JSP to be compiled and thus become the owners of the .java and .class files that result.

# ASF Tomcat log files



- The log files are all based on the configuration in the server.xml file
- The default location is in the /logs directory under the tomcat\_home directory
- The log files include:
  - jasper.log
  - servlet.log
  - tomcat.log
  - jvmstderr.txt
  - jvmstdout.txt

# Notes



ASF 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. These log files are all specified on the configuration file `server.xml`. Those files are located in `/logs` directory under the `tomcat_home` directory by default. Under this directory structure you will find the files shown in following table.

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> obj
<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>





# Performance

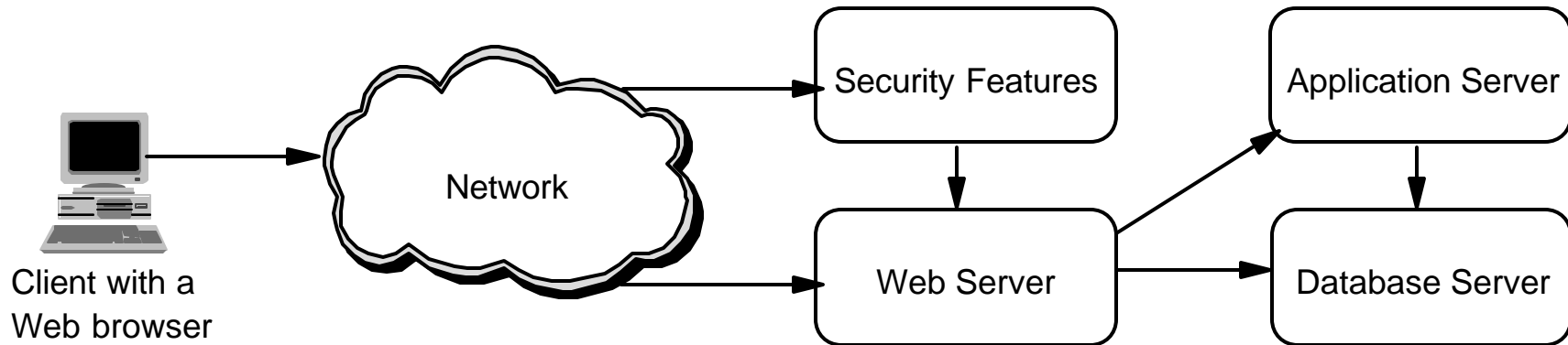
# Notes



There are several things that affect web server performance. The HTTP Server (powered by Apache) has some performance related values, functions that provide cache mechanisms and cache manager. We will introduce these values and functions to you in this part.

- ▶ Web performance components
- ▶ Web server performance
- ▶ Global performance values
- ▶ Specific performance values
- ▶ Local cache
- ▶ Triggered Cache Manager (TCM)
- ▶ FRCA: Features
- ▶ FRCA: "Normal" and local cache
- ▶ FRCA: Cache miss scenario
- ▶ FRCA: Cache hit scenario
- ▶ FRCA: Limitations
- ▶ FRCA: Configuration
- ▶ FRCA: Collection service

# Web Performance Components



- Client

- Processor speed
- Memory
- Hardware
- Browser

- Server

- Web server
- Application server
- Database server
- Security features
- Caching

- Network

- Routers
- LAN topology
- Link speeds
- Packet filters
- Proxy and Proxy caching
- Socks servers

# Notes



There are three major components of a Web server environment, each one with its own performance requirement and limitations.

Those 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 could 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.

A problem in any of these areas may impact the performance of your Web application.

Our focus here is on the iSeries Server. The client and network components do directly impact the Web server performance and we will provide a brief description of each component's impact the overall performance.

## 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 as many Web-related tasks use large amounts of memory to be complete. If the client does not have enough memory, the user could perceive performance problems because of the client configuration.

**Hardware-** Every piece of hardware is important. Hard disk and communication adapters are important when the performance is an issue. Keep in mind that clients not are updated as fast as the IT technologies change.

**Browser-** Since Web browsers are the main interface in Web server, browsers must be updated frequently. Some Web servers, customer applications and Web application servers rely on browser capabilities.

# Notes



## The network

Usually the network has more impact on overall performance. 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 if we identify some network components:

- Routers
- LAN topology
- Link speeds
- Packet filters
- Proxy and Proxy caching
- Socks servers

As there are many components involved, you can spend a lot of time trying to get 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. Even more, many of those components could be completely outside your zone of responsibility (for example, the Internet).

## The server

The server behavior is impacted by several factors including application, resources and database components. Each scenario has his own components. You should 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. The figure above shows server components involved in most of the e-business implementations.

These components can all impact server performance as most of the time a client request requires processing in each one of those components.

Other iSeries internal features (for example memory, bus, and disk) must be analyzed for your Web application too because if the iSeries server itself does not has the internal resources to handle the requirements performance will be impacted.

# Notes



# Test Results: Hits per Second (est)



Transaction Type	Non Secure Trans/sec per CPW	Non Secure Trans/sec per CPW	Secure Trans/sec per CPW	Secure Trans/sec per CPW
	V5R2	V5R1	V5R2	V5R1
Static Page (IFS, non-cached)	1.92	1.51	1.52	0.69
Static Page (cached)	3.10	2.05	2.23	0.82
Static Page (FRCA)	15.29	not available	not applicable	not applicable
CGI - new activation	0.06	0.08	0.06	0.08
CGI - named activation	0.43	0.40	0.41	0.31
Persistent CGI	0.33	0.32	0.30	0.26
Net.Data	0.19	0.22	0.18	0.20
WebSphere Servlet	0.77	0.49	0.37	0.34
User Module	3.03	1.21	2.23	0.62

- IBM HTTP Server Powered by Apache or iSeries
  - V5R2: WebSphere Application Server 4.0.2; 100 Mbps Ethernet
  - V5R1: WebSphere Application Server 3.5.3; 100 Mbps Ethernet
- Based on 270 with moderate web server load measurements
- Data assumes no access logging, no name server interactions, Keep Alive ON, LiveLocalCache OFF
- Secure testing: 128-bit RC4 symmetric cipher and MD5 message digest with 1024-bit RSA public/private keys
- CPWs are "Relative System Performance Metrics" used to rate AS/400 and iSeries servers
- Web server capacities are base on total CPU available. Capacities may not necessarily scale by CPW. Actual results may differ significantly.
- Transactions using more complex programs or serving very large files (pages) will have lower capacities shown in this table

# Notes



This table is an excerpt from the V5R2 Performance Capabilities Reference manual. This manual discusses the internal IBM benchmarks (CGI program, Net.Data and so forth transaction processing and also has some test results showing how larger file (web page) sizes can further lower the CPW multiplier. Other Performance Capabilities Reference manual information also shows an alternate way of estimating hits per second capacities of an iSeries - CPW required per transaction per second estimated values.

Review the notes and disclaimers on this foils. The numbers shown assume the entire processor (CPW capacity) is available for web serving.

Important considerations include:

- V5R2 test results showed mostly improved (higher CPW number the more hits per second estimated) CPW multipliers for static (read only, no program processing) and WebSphere servlet and User Module (uses Apache APIs) transaction types.
- Use of V5R2 **Fast Response Cache Accelerator for unsecured static (not changed) data shows a significantly higher CPW multiplier rating.** Please note that FRCA does not apply to SSL data.
- Examples using this table are:
  - Assume a 270 of Processor CPW rating 1070 (# 2432) with 100 percent of the processor rating is available for HTTP web servings. Assuming your application is similar to the IFS, non-cached transaction tested, you multiply 1.92 times 1070 and get an estimated number of hits per second value of **2054 transactions** (hits) per second.
  - Assuming the same full processor rating of 1070 and same non-cached transaction, but with only 20% of the system available for web serving, you multiply 1.92 by 1070 by .20 to get **410 transactions** per second.
- Follow the performance tips in the Performance Capabilities Reference manual for optimal performance.



# 2002 iSeries Benchmark Results



Benchmark	iSeries Results	Status	Significance
VolanoMark (Java)	<b>1st Overall</b> 283, 000 iSeries 890 32-way	Published 6/16/2002	Re-establishes iSeries as leading Java server. Demonstrates scalability of i890 in Java - over twice the throughput of iSeries 840 running V5R1
Intentia Movex (Java)	<b>1st Overall</b> 2.38M trans/hr iSeries 890 32-way 2X Improv	Published 6/16/2002	Demonstrates iSeries 890 transaction processing scalability through dominance in this leading ISV Java application benchmark. Once again, 2X improvement over iSeries 840 running V5R1
Notesbench-Mail	<b>1st Overall</b> 150,000 Users iSeries 890 32-way	Published 8/29/2002	Puts iSeries back in the lead of this key industry benchmark, besting previous best of breed by 40%. Demonstrates leadership performance for the non-Intel marketshare leader, iSeries.
SPECweb99 (e-Commerce)	<b>1st Apache</b> (3rd overall) 12,900 hits/sec iSeries 890 16-way	Published 9/03/2002	1st time iSeries has ever appeared in this critical industry benchmark, and in a leadership position as the only Apache-based webserver in the list. Demonstrates tremendous performance improvement in V5R2 web serving capability and puts iSeries on the web serving map.
SPECweb99SSL (secure e-Commerce)	<b>1st Overall</b> 3,600 hits/sec iSeries 890 16-way	Published 9/03/2002	iSeries LEADS in this industry benchmark, asserting its unparalleled SECURE ebusiness scalability. Results are over two times better previous best of breed, and demonstrate that iSeries is the best webserver for iSeries customers to use.



# Notes

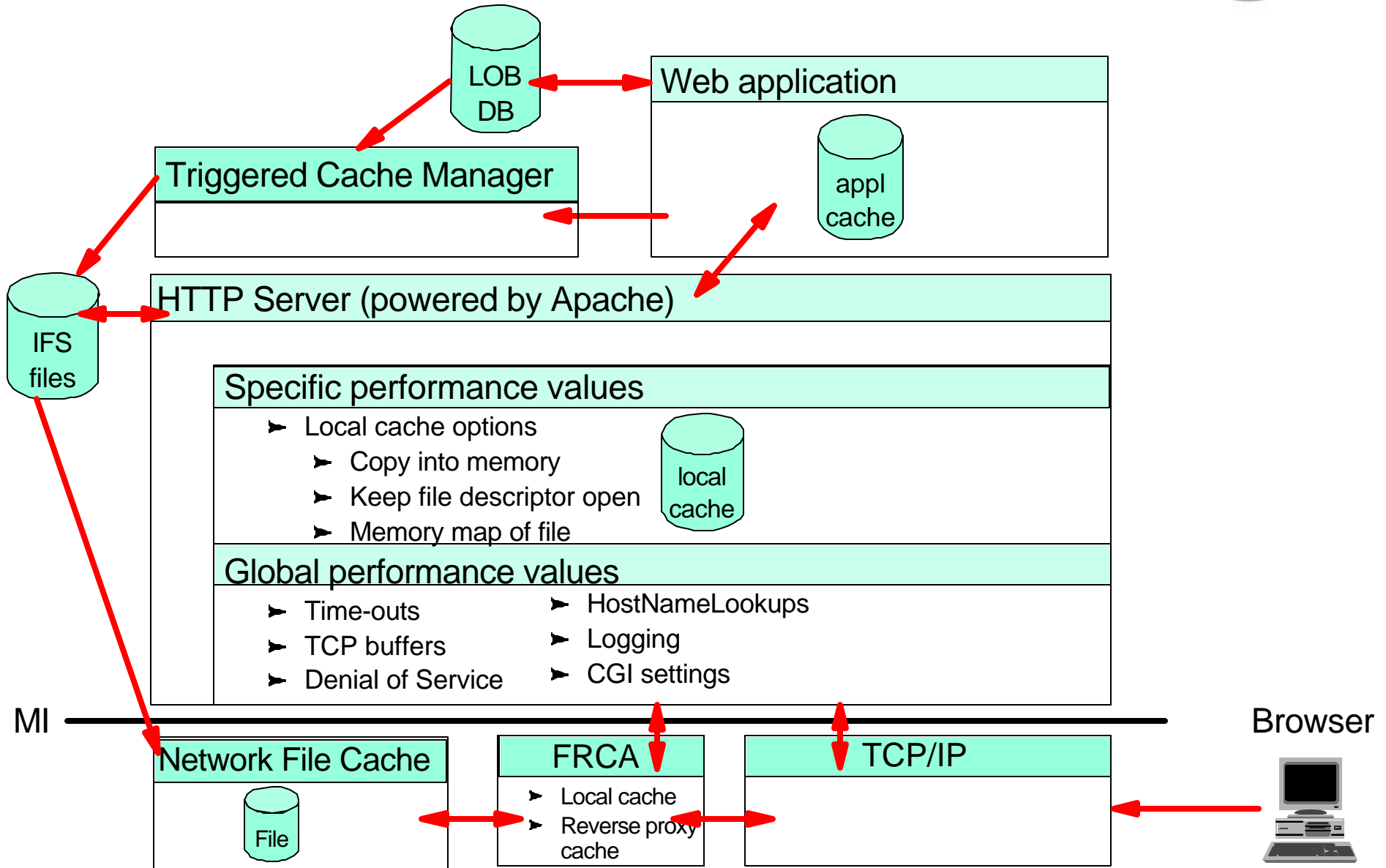


The comment to be made is 'wow'. As we should expect the iSeries leads the way with fantastic performance.

Also, the other competitors in the SPECweb99 space are using not the Apache server, but other smaller Web servers that are maybe better tuned to SPECweb99 than to actually serving real customer environment. This is why we make the point that the iSeries is tops in the Apache server.

Other non-Apache servers that the competitors might use would be Zeus and TUX. Of course, IIS and Red Hat Content Accelerator would also be seen in the charts.

# Web Server Performance



# Notes



There are several things that affect your server's performance. The HTTP Server (powered by Apache) include a set of internal components that allow us to improve the Web server performance to process the client request in a fast and reliable way. Those server components can be used with any type of data served by the HTTP server.

Using the HTTP Server (powered by Apache) we can improve the Web server performance at two different levels:

- Using global parameters that allow us to configure the attributes used by all the HTTP servers in your iSeries server.
- Using specific parameters based on the type of data the client is requesting. These specific parameters generally revolve around the concept of a local cache, proxy cache, or the FRCA (Fast Response Cache Accelerator) used by the HTTP Server (powered by Apache).

In addition, using a Triggered Cache Manager (TCM) server can dramatically improve the performance of your Web application by being proactive in creating dynamic web content and placing it in the iSeries Integrated File System (IFS) to be served at static document speeds.

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 traffic. To this end the iSeries provides a set of APIs as seen in the part of "High Availability" .

The figure above shows the Web server components available to improve the performance using the HTTP Server (powered by Apache). When the client sends the request, configured global performance values are used to optimize the work performed by the HTTP Server (powered by Apache) Web server. The Web server then tries to process the request either using data cached in the local cache (memory) or to serve the file from the IFS.

If the content is to be dynamically generated, control is passed to a Web application that can optionally access Line of Business (LOB) databases to *reactively* create content.

Changes to the LOB database or other 'triggering' mechanism can cause the Web application to be proactive in the update of pages directly in the iSeries IFS. In this way, even before the first client request arrives at the iSeries Server the content (a web page with dynamic data pulled from the LOB database) can be generated and stored in the IFS as a 'static' document to be quickly served by the HTTP server.

Clearly, the sooner cached data can be used during client request the fewer resources and time that is needed for the entire end-to-end transaction between client and server.

**FRCA (Fast Response Cache Accelerator):** is an IBM research cache based on a software architecture (AFPA) that dramatically improves the capacity/performance of Web and other TCP servers. FRCA, in V5R2 (only), is used for the HTTP Server (powered by Apache) only. The architecture includes a network file cache that serves non-secure static (via a local cache option) and dynamic (via reverse proxy) content directly from beneath the MI. More on FRCA later in this presentation.

# Global Performance Values



**Threads and asynchronous I/O**

**Process control: HotBackup**

**Logging**

**HostNameLookups**

**KeepAliveTimeout**

**TCP buffer size**

**Denial of service**

# Notes



Global parameters are the parameters that 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.

## 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 QHTTSPVR 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 maximum number of threads found on the Global Server Settings form. Directive: `ThreadsPerChild`.

You can set this parameter at the Global Server Setting to be the default value at start up for all your Web Servers (original and powered by Apache). Then, you have the option of overriding this Global Server Setting for each HTTP Server (powered by Apache).

You can *only* configure the maximum number of threads the server opens at start up. The HTTP Server (powered by Apache) will always start with the maximum number of configured threads. The HTTP Server (powered by Apache) implementation does not use the minimum thread value.

When there are no available threads the Web server response time, from the client point of view, is impacted as the request takes longer because the lack of available threads. Setting this number too low impacts the server performance as the client request can not be process 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 I/O model, threads are only involved in processing when there is work to be done. Threads are dispatched to perform work as required and 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 utilizing 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. We recommend you keep the default value of on (or enabled).

Directive: `AsyncIO`.

# Notes



**Tip:** Asynchronous I/O is one of many enhancements to the standard Apache server as delivered to IBM Rochester by the Apache Software Foundation. This is just one of the many reasons that the parenthetical (powered by Apache) means integration.

## Process control: HotBackup

The HotBackup directive is used to specify whether or not 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 will immediately take over and act as the primary and continue 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 take over and the end users do not detect the loss of server. However, some HTTP requests in transit 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.

This setting can be configured using **Process Control** in the admin GUI.

**Tip:** An example of a loss of network might be if one of two interfaces on the iSeries fail. The routes bound to the failing interface would cause all the connections across that interface to also fail.

An elegant solution to this potential problem would be to configure a virtual IP address (or a circuitless connection) which is an IP interface that is defined on the system without being associated with a physical hardware adapter. These addresses can always be active on the system. If, for example, one of two physical interfaces fail then all network traffic can be re-routed through the active interface and the applications and HTTP server will never know of the problem. For more information about using virtual IP addresses see V4 TCP/IP for AS/400: More Cool Things Than Ever, SG24-5190.

## Logging

Logging is another setting that impacts server performance. Logging here applies to the Error Log only.

Simply put, as you request a higher logging level, 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 will increase and the Web server performance (may) decrease.

This parameter can be configured for every HTTP server and for each virtual context within the the HTTP server. If the HTTP server has a different Error Log file for every virtual host context, you should consider that a file descriptor is opened for each log file. Opening too many descriptors can impact system performance.

This setting can be configured using **Error Logs** in the admin GUI.

# Notes



## 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 Include (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 might make it easier to track the usage of your web site (by geography, for example) or determine problems.

The default for this directive is off to save on the 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 considerable amounts of time and resource.

This setting can be configured using the `HostNameLookups` directive.

## KeepAliveTimeout

This setting is used to control whether or not the Web server works with persistent connections. Persistent connections enables 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 the server waits for subsequent requests before closing the connection. If this value is too low, the server could be impacted in terms of performance as connections could be closed frequently. If this value is too high, the Web server could have many connections open and the server could run out of resources. In this case the use of asynchronous I/O can alleviate (but not eliminate) the problem of running out of resources.

This value applies to each client request. If the Web server you are working with is used in Internet, keep in mind that as this is a communication setting, the value you select here could be correct for some environment but not for others. We recommend leave the default value unless you have studied your environment and have reason to make a change. One might be if you know that persistent connections are not supported all the way between the client and your server.

This setting can be configured using **HTTP Connections** in the admin GUI.

## TCP buffer size

The TCP buffer size attribute provides a limit on the number of outgoing bytes that are buffered by TCP. Once 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, which 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.

This setting can be configured with the `SendBufferSize` directive or using **Performance** in the admin GUI.



# Notes



## Denial of service

The denial of service attribute is equally a performance setting as well as a security setting. This setting allow us 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 to the one it expects. Although this setting impacts the server performance as each request is tracked, it allow 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) include 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 (0) which indicates there is no maximum size specified. Directive: `LimitRequestBody`.

**Maximum XML message body size-** Allows you to limit the size of an XML-based request body. The default value is 1000000 bytes. Directive: `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. Directive: `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. Directive: `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. Directives: `LimitRequestLine`.

The denial of service settings can be configured using **Denial of Service** in the admin GUI.

**Tip:** This built in protection against various denial of service attacks is one of the many integrated extensions to the standard Apache Web server. Again, it is one of the reasons for the (powered by Apache) parenthetical in the HTTP Server (powered by Apache) formal name of the server on the iSeries.

# Notes



# Specific Performance Values



**Local cache**

**Triggered Cache Manager (TCM)**

**Fast Response Cache Accelerator (FRCA)**

# Notes



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 might declare that your home page is static. After all, it is made up of just HTML, a bit of Java script and GIFs. But even your home page will change from time-to-time. And 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.

But, to ease our understanding we do need to define content that is not changing all that often is considered static and that content that is changing based upon database information and user input is considered 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 three caching mechanisms:

- Local cache
- Triggered Cache Manager (TCM)
- Fast Response Cache Accelerator (FRCA)

**Tip:** The TCM is not a cache but a cache manager. In effect the TCM will help 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) will serve them from the IFS like static content.

# Local Cache



## Local cache implementation

- Define the memory size for the files to be cached
- Define the cache method

## Cache methods:

- Copy into memory
- Keep file descriptor open
- Memory map of file

## Directives for cache option

- LiveLocalCache
  - Dynamically invalidate local cached files when they are updated in the IFS
- DynamicCache
  - Dynamically add new files to the local cache

# Notes



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 and so on. 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 pre-loaded in the server's memory at server start up and the amount of memory used for this purpose.

The local cache implementation is a two stage process:

1. You define the memory size for the files to be cached.
2. And then you 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 start up time and any changed or new files cached due to dynamic caching . The server directive to identify the memory size is `CacheLocalSizeLimit`.

The files can be cached at server startup using any of these three methods:

- Copy into memory
- Keep file descriptor open
- Memory map of file

## Copy into memory

The copy into memory method allow us to define the file or files that will be pre-load in the iSeries memory, in the memory pool used by the HTTP server at Web server start up. The memory size can be setup according to your application requirements. There is no limit for the memory size used to pre-load 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 allow us to define ASCII file or files whose descriptors are cached at server start up. 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`.

# Notes



## Memory map of file

The memory map of file method is similar to the Copy into memory method, but the difference here is that this method uses a memory pointer to specify files that should be cached at start up, 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 very powerful pair of options gives your HTTP Server (powered by Apache) server the ability to:

- ▶ Dynamically update the (static) files that were placed in the local cache at server start up time. The default value is on (or enabled)  
This directive (`LiveLocalCache`) will check to see if the file has been updated in the Integrated File System (IFS) each time it is requested. If not, the file is served from the cache. If it has, 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 would have to restart your server to cause it to be loaded back in the local cache.

If `LiveLocalCache` is off then your HTTP Server (powered by Apache) server will not check to see if 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 would be very useful in a directory of all GIFs, for example, that rarely change.

- ▶ Dynamically add new (static) files to the local cache based upon demand. The default value is off (or disabled).  
This directive (`DynamicCache`) allows dynamic caching of (static) files. Because of the overhead involved in determining if a file being served should be added to the cache impacts all the files being served from your Web server - we would recommend using this directive for sites that generally have less than 1000 files.

Or, maybe to put this another way, is that 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 start up time - then you may try `DynamicCache` on. But, it will always be better to identify all the files you want to add to the local cache at server start up time as this is far more efficient.

The dynamic cache will only add files to the local cache as long as there is still room as defined by the directive `CacheLocalSizeLimit`. If the local

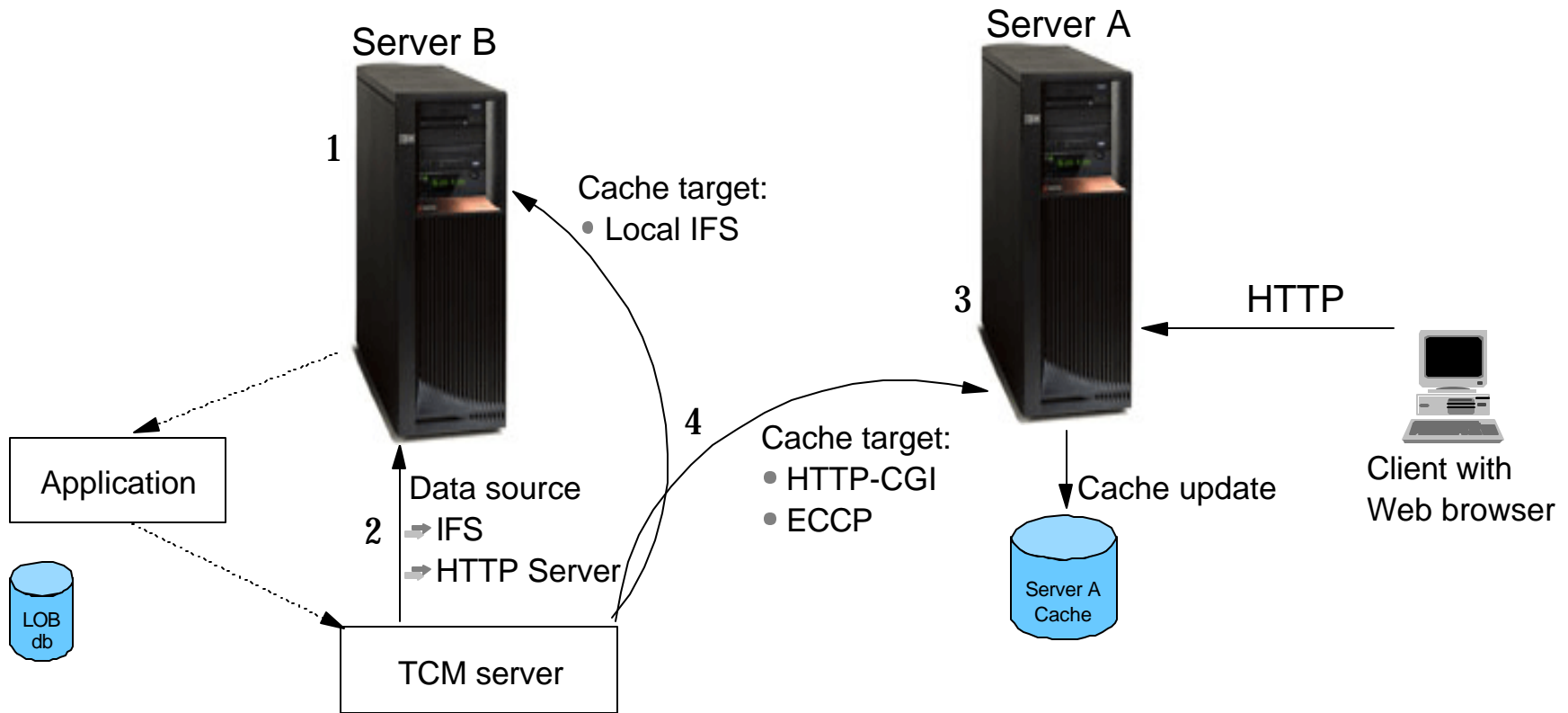
cache is full no more files will be added to the cache.

# Notes





# Triggered Cache Manager (TCM)



# Notes



The Triggered Cache Manager (TCM) is a component in the iSeries server that was created exactly for this purpose. This component is packaged in the IBM HTTP Server for iSeries, 5722-DG1, option 1.

The TCM is a new TCP server. This server may be used in conjunction with web servers and web document caching agents to keep web sites running at peak performance.

## The Triggered Cache Manager:

- Is a cache manager, not a cache or cache server.
- Works based on trigger messages, which means you have to set up application triggers for the server to work.
- Is 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 have described here for the HTTP Server (powered by Apache) but could be used for the HTTP Server (original) or a Domino environment as well.

What TCM does is to be proactive (based upon updates to a LOB db) in the update of Web content that looks static to the 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. An example of this might be a Web site that serves an on-line catalog which contains price and inventory information.

One of IBM's first uses of the TCM concept was to drive the 1996 Summer Olympic Games Web site. Think of a 'results page' 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 (Server B) had to re-run all the db queries to dynamically calculate the content that would likely bog down the server with many repetitive actions. On the other hand, if only when the application LOB data (see Figure 8-14) was updated with new results then the application server would update a standard 'results page' to be served from a 'static' portion of your web site (Server A) - that would cause a tremendous reduction in the burden on the application server (Server B).

**Important:** In the figure, it should be noted that Server B and Server A can be the same iSeries server - if and only if the mechanism to update the cache target used is the local IFS or CGI. That is, the iSeries server does not support being a cache target using the External Cache Control Protocol (ECCP). The iSeries server also does not provide a CGI application to handle the cache update requests. You may, however, create your own CGI application to handle cache update requests sent from the TCM.

# Notes



Here are some components we should identify in the figure:

- 1 The Web server where the data is located. In our case an iSeries server.
- 2 The mechanism used to retrieve the data.
- 3 The server used to cache the data. Server A could be the same server as the iSeries Server B.
- 4 The mechanism used to update the cache data. Three are identified. Local IFS allows the TCM server running on your iSeries to directly update files in the IFS. To update servers that are network connected to the TCM server you must use either HTTP with a special CGI program or ECCP. ECCP is used to update web document caches on IBM model 2212 and 2216 network routers.

For example, the application running on Server B uses as the LOB db a table of items. This table includes the item code, item description and item price. One of the most common request, is an HTML Web page that contains the entire item catalog.

As this information is accessed so frequently, we decide to proactively generate this document and copy it into the IFS on Server B.

When the item table in Server B is updated with a new item, either a database trigger or the application sends a trigger message to the TCM server. The TCM server then identifies what content must be updated and updates the document as necessary.

Identifying the content that should be cached requires a good knowledge of the application. Because of the nature of some applications, for example, a banking application, the information requested by each client is different and there might not be a reason to cache some data.

## How TCM works

There are three basic steps involved before and during the TCM process:

1. Monitor for data changes.
2. Send a trigger message to the TCM server with the information about the data that was changed.
3. Allow TCM to request the dynamic page from the application server
4. Allow TCM to update the dynamic page content stored in the IFS or other cache targets.

The way TCM determines which pages need to be updated, is consulting an Object Dependency Graph (ODG). This ODG is a data repository used to store dependency relationships for the pages the server handle. Thus, when the data changes, TCM find out what pages are dependant on the changed data. Once 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 allow then, the dynamic update of the caching without any server restart.

There are multiple scenarios where you can use TCM. One for example could be an HTTP server with one TCM server, other could be multiple HTTP servers with multiple TCM servers. Or any other combination between HTTP and TCM servers.

# Notes



# Fast Response Cache Accelerator



**A software architecture that dramatically improves capacity of web servers**

- Adaptive Fast Path Architecture

**Industry leading Web server performance**

- iSeries and pSeries currently leading SPECweb99 using FRCA technology

**Core technologies**

- In-memory Cache
- Reverse split-connection proxy
- Layer-7 (application level) router

**Built in kernel (SLIC) for performance**

**Implemented across IBM platforms**

# Notes



## Implementations:

Windows 2000

- First implementation of AFPA by IBM Watson Research

z/OS (OS/390)

AIX (pSeries)

- world record in SPECweb99 benchmark (21,000 simultaneous connections)

Linux

- Plan Open Source release

iSeries 400

- V5R2 HTTP Server (powered by Apache) only

# Notes



## Layer-4 versus Layer-7 Routing:

	<u>Layer Four Routing</u>	<u>Layer Seven Routing</u>
<u>Layer:</u>	Transport (TCP)	Application (HTTP)
<u>Information:</u>	Source, destination IP address, TCP port	HTTP headers (URL, method, "Host:", etc.)
<u>Connection:</u>	One connection from client to server	Two connections, client-router, router-server
<u>Servers:</u>	All content, function on all servers	Can partition content, function across servers
<u>Advantages:</u>	Efficiency, routes all applications (HTTP, FTP, telnet, etc.)	Flexibility, can separate static, dynamic content

# Notes





# FRCA: Features



## Two new components that work together

- Fast Response Cache Accelerator (FRCA)
  - Provides system API set and framework for socket applications
  - Accelerates file serving performance for the HTTP server
  - The *one V5R2* example is the HTTP Server (powered by Apache)
- Network File Cache (NFC)
  - Provides SLIC level cache

## Configurable by new FRCA directives in Apache server config

- Can be enabled for each listen port
- Local cache: Specify file name (with wild cards) for "static" content caching
  - When content is updated NFC automatically uses new file
- Reverse proxy cache: Specify URI for "dynamic" or remote content caching
  - Timer used to determine when cached items are stale

# Notes



Responding to the growing need for improved speed and performance of Web servers, IBM research has defined the Adaptive Fast Path Architecture (AFPA). AFPA has been implemented on several server platforms including Windows NT and Windows 2000, OS/390, AIX and most recently Linux. The external product name is most commonly known as Fast Response Cache Accelerator (FRCA).

AFPA is a software architecture that dramatically improves the capacity of Web and other TCP servers. The architecture defines interfaces that allow these generic mechanisms to be exploited to accelerate a variety of application protocols, with the focus on HTTP. The architecture is general purpose and applicable to many TCP servers, including FTP, NFS, DNS and Domino.

For OS/400 V5R2, this architecture is implemented as the FRCA feature with the HTTP Server (powered by Apache).

Since, the iSeries TCP stack runs in a SLIC router task and not a software interrupt, the FRCA code also executes in a SLIC router task context. The SLIC based implementation eliminates the overhead of switching from a SLIC router task to a user-level server thread. FRCA provides system APIs that can be used by system applications, at this time, the HTTP Server (powered by Apache). HTTP Server (powered by Apache) uses this APIs to work with SLIC FRCA code and Network File Cache (NFC) for serving contents.

The AFPA architecture includes a network file cache that serves non-secure static content. This architecture is implemented as the Network File Cache (NFC) with FRCA feature for the HTTP Server (powered by Apache). NFC provides the capability to efficiently store and retrieve cached entries of file data and user data.

FRCA directives are provided to the HTTP Server (powered by Apache) that enables your HTTP server to use FRCA cache. FRCA cache can be enabled for each listen port in the server configuration. This allows us to make choice if you use FRCA cache or not for each Listen on a specific <IP address:port>.

Static content can be cached by specifying file name using certain directives. The loading to the cache occurs during starting up of the HTTP server or the first access to that file, and this depends on which directive is used. You can use an asterisk (\*) as a wild card character on the file names. Then you can specify the directory name with the asterisk to make FRCA cache for some temporary files like the ones created by TCM.

Dynamic content such as result of CGI or servlet can be cached by specifying URI of the request and cache life time. This is a reverse proxy cache support that allows you to access an HTTP server either on this same iSeries or anywhere on your intranet or Internet.

# Notes



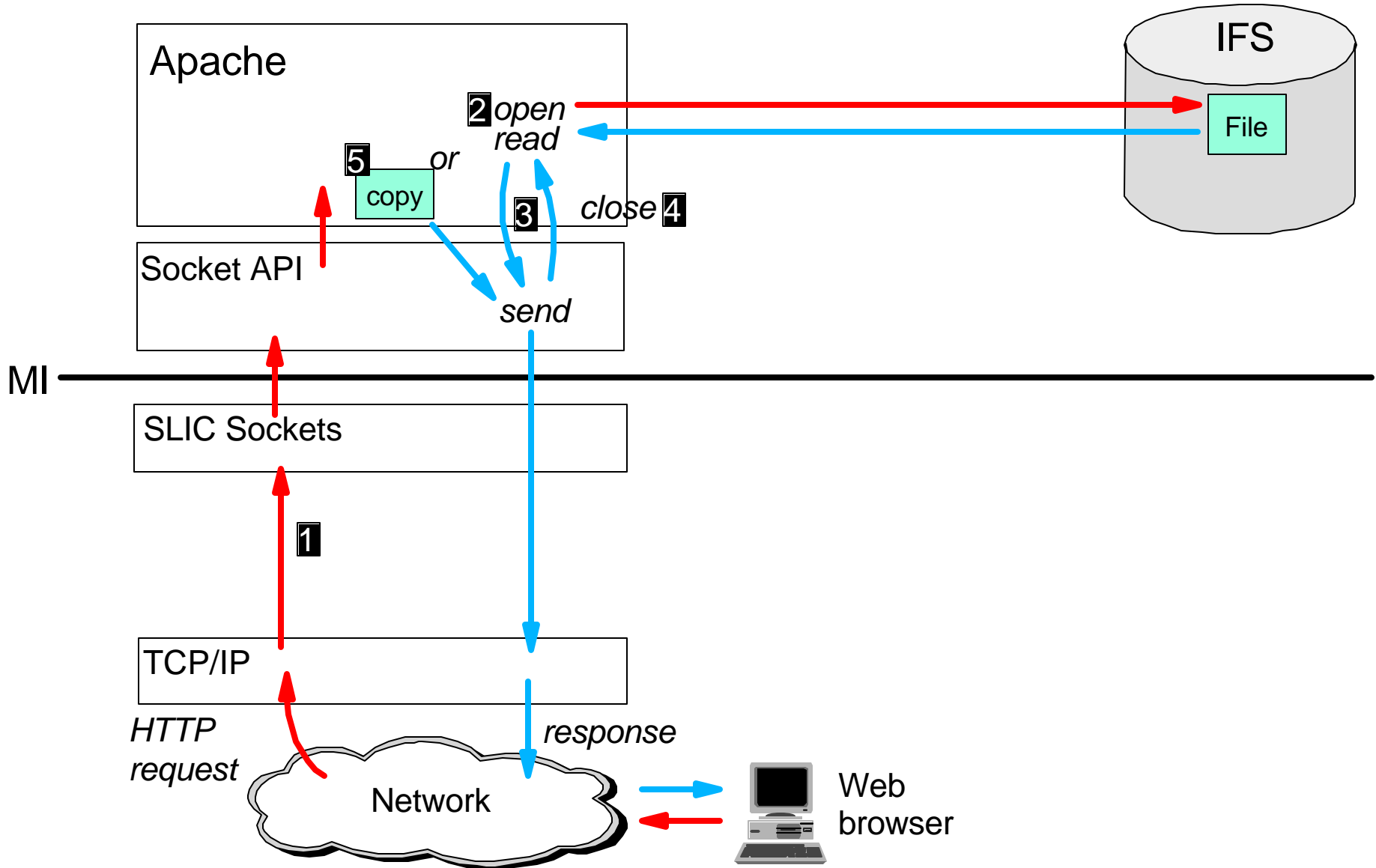
Forward vs. Reverse Proxy Caches:

	<u>Forward Proxy Cache</u>	<u>Reverse Proxy Cache</u>
<u>Location:</u>	Close to <b>clients</b>	Close to <b>servers</b>
<u>Design:</u>	Optimized to cache "entire Internet"	Optimized to cache specific web servers
<u>Browser:</u>	Browser may be configd. to use proxy	Reverse proxy is transparent
<u>Scenario:</u>	ISP wants to reduce bandwidth required to support N <b>clients browsing</b> Internet	ISP wants increase capacity of <b>servers</b> hosting web sites

# Notes



# FRCA: Local File Serving before FRCA



# Notes



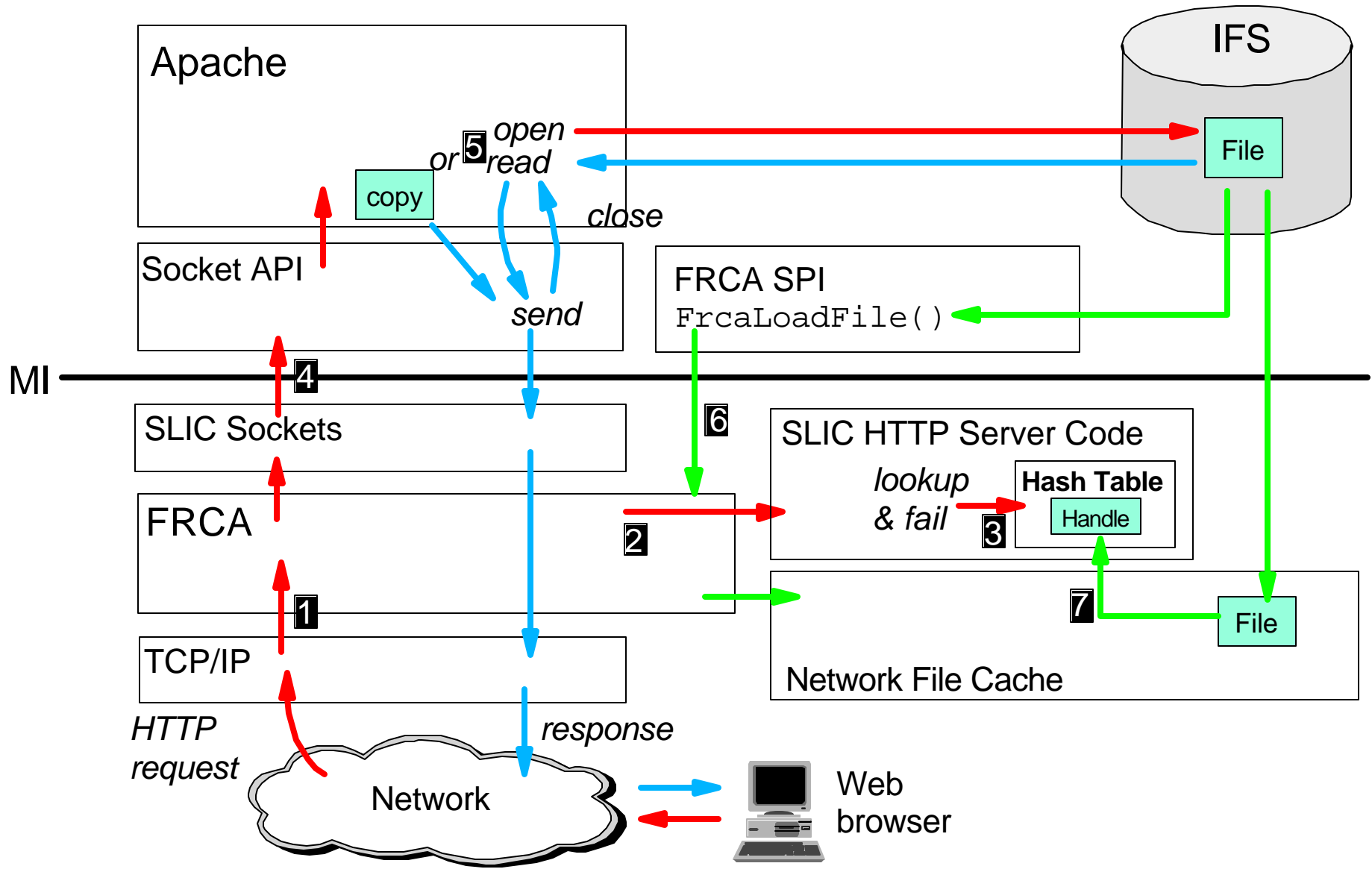
Now we will see how the FRCA works.

First of all, we will show you the "Normal" behavior of file serving performed by HTTP server without the influence of FRCA. At this time a local cache may be used.

The processes are as follows:

- 1 An HTTP request is received by TCP and passed to the Apache Web Server using sockets interfaces.
- 2 The Apache Web Server parses the HTTP request and maps the URL to an IFS file and opens it.
- 3 The Apache Web Server reads the file to build the HTTP response and calls Sockets send() to send the HTTP response.
- 4 After sending all the data, the Apache Web Server closes the file.
- 5 When the file is found in the local cache, the action of open, read and close are eliminated from the steps and the copy of that file from the cache is sent using Sockets send().

# FRCA: Local Cache Miss Scenario



# Notes



Next, we will show you the behavior how the FRCA loads the file into the NFC when it misses the file in the NFC cache. That is, the cache miss scenario.

The steps are as follows:

- 1 An HTTP request is received by TCP and passed to the FRCA.
- 2 The 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 HTTP logical cache (Hash table, one per server instance).
- 4 When the HTTP logical cache lookup fails, the HTTP request is punted (redirected) to the Apache Web Server using normal sockets interfaces.
- 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() SPI is called to load the file in the NFC. Note, the Apache Server ensures that only files with public access are loaded. Parameters supplied on the FrcaloadFile call include, file descriptor (file must be open), URL search key and HTTP header information. The FrcaloadFile is passed through the control pipe to the FRCA code.
- 7 The FRCA calls NFC to load a single copy of the file in the Network File Cache. A NFC handle returned by NFC, the URL search key and HTTP header information is passed to the SLIC HTTP Server code and added to the HTTP logical cache. Note, only a NFC handle is cached by the SLIC HTTP Server, and not the entire file data; a single copy of the file is managed by NFC.



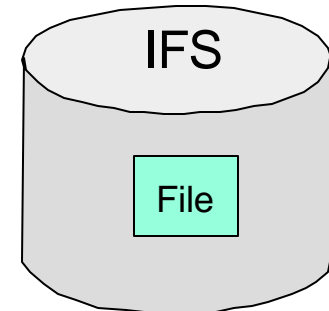
# FRCA: Local Cache Hit Scenario



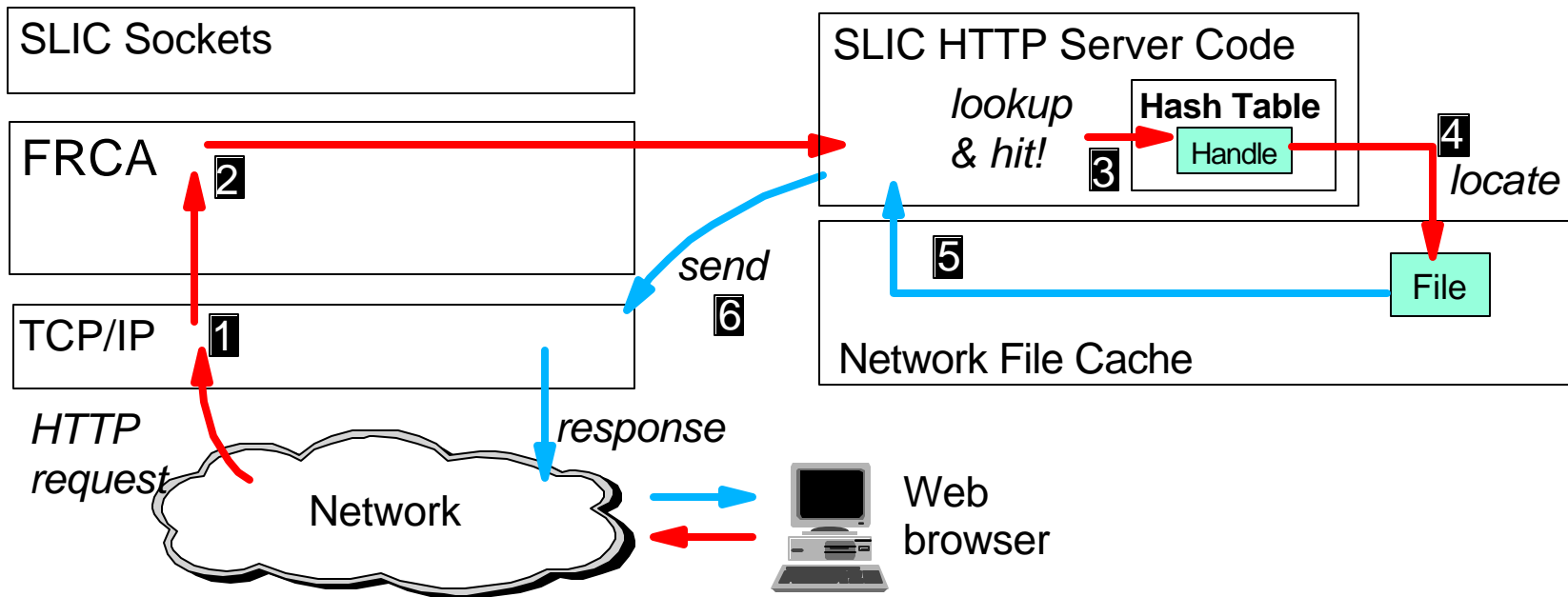
Apache

Socket API

FRCA SPI  
FrcaLoadFile()



MI



# Notes



The figure above shows the scenario when the file is found in the FRCA cache. That is, the cache hit scenario.

The steps from request through response are as follows.

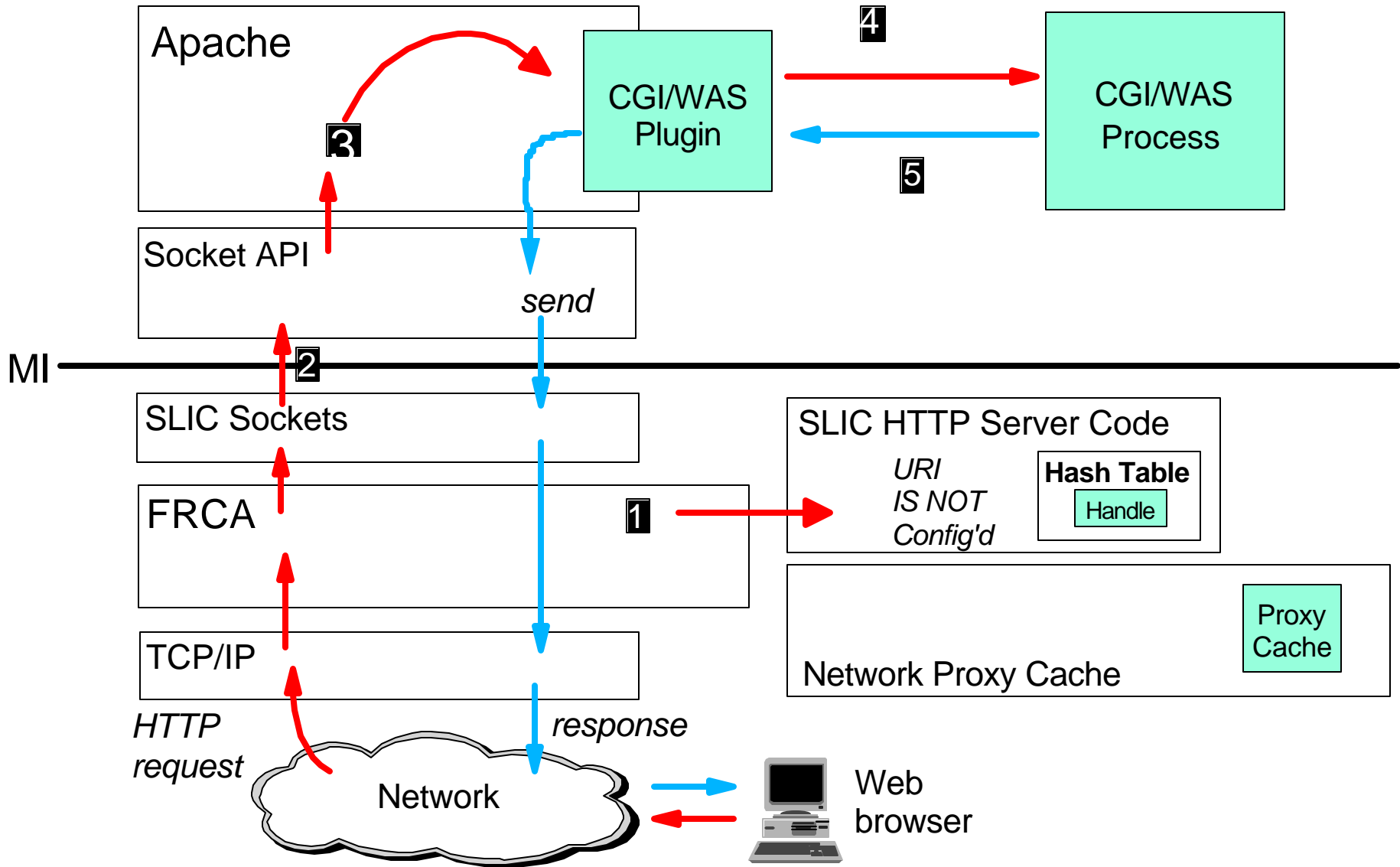
- 1 An HTTP request received by TCP and passed to the FRCA.
- 2 The FRCA intercepts the HTTP request and passes it to the SLIC HTTP Server code.
- 3 The SLIC HTTP Server code parses the HTTP request and uses the URL as a search key into the HTTP logical cache (Hash table).
- 4 When the HTTP logical cache lookup is successful, Network File Cache (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 and links the file data to it, and sends it as a response through TCP/IP.

The above path can result in a sizable performance (of a single transaction) and capacity (allowing more transactions per unit time) improvement due to:

- Never having to go above the MI. This results in:
  - No task switches to the threaded job model above the MI
  - Could (depending on a variety of things) save 2 copies of the data. FRCA will not copy the data it finds in the NFC. FRCA will directly send the data to the TCP/IP stack in the iSeries SLIC.
- Code path length should be shorter which will result in CPU utilization for cache hits to be lower.

Details from the V5R2 Performance Capabilities Reference are coming up later in this section.

# FRCA: Content Scenario



# Notes



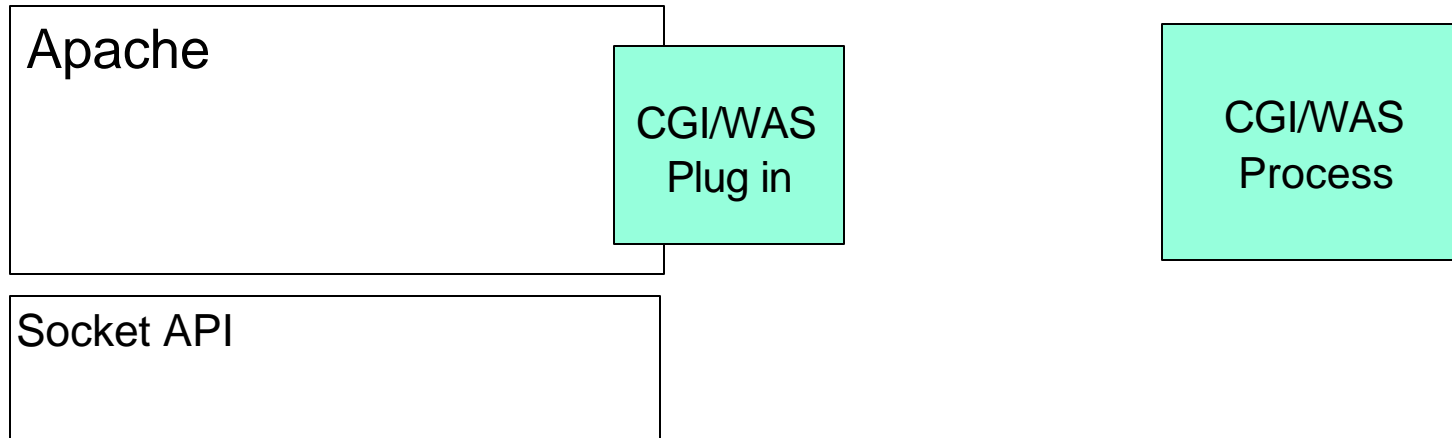
The figure above shows the a general dynamic content scenario. That is, when the HTTP Server (powered by Apache) needs to get some dynamic content from some content server. This is the basis for FRCA's reverse proxy caching.

The steps from request through response are as follows.

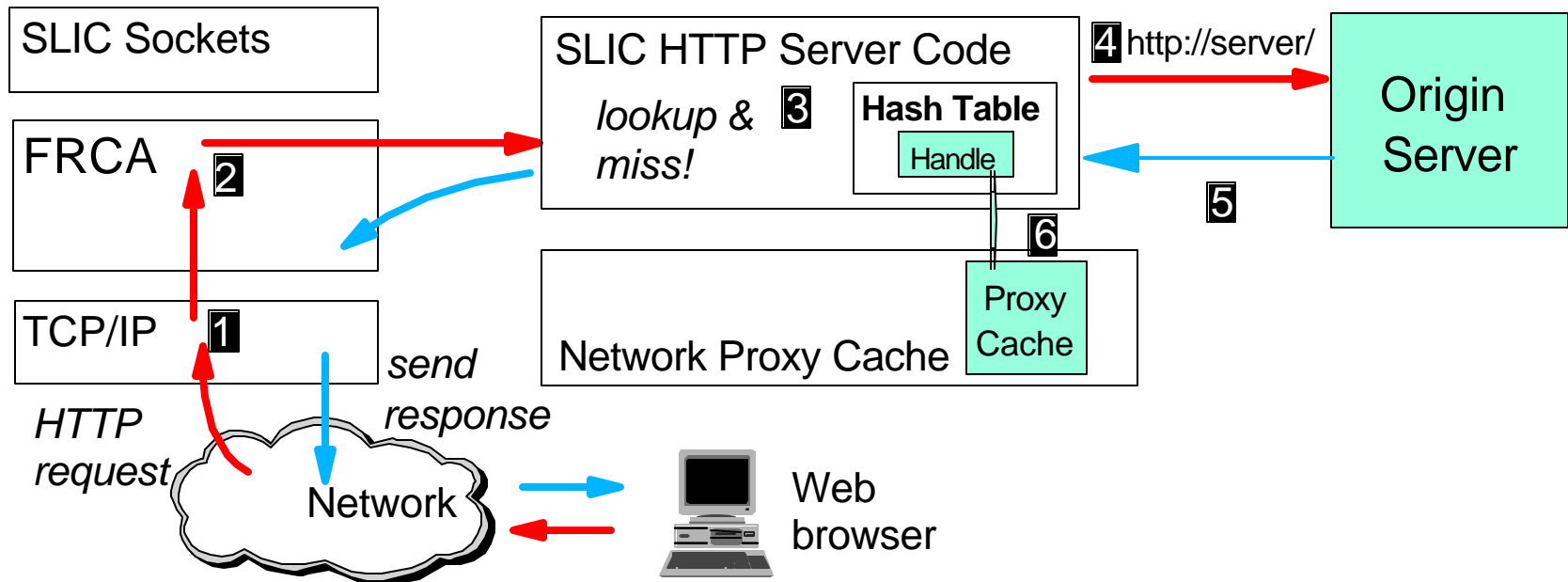
- 1 An HTTP request is received by TCP and passed to the FRCA. FRCA sees that the incoming URI is not configured for reverse proxy cache.
- 2 When the HTTP logical cache lookup fails, the HTTP request is punted (redirected) to the Apache Web Server using normal sockets interfaces.
- 3 The HTTP Server (powered by Apache) parses the HTTP request and evokes the configured plug-in. This is business as usual for the HTTP Server (powered by Apache).
- 4 The dynamic content server is called.
- 5 The dynamic content server returns the content that is sent back to the browser.

**Note:** You will notice that the NFC has been replaced in the graphics associated with FRCA reverse proxy caching with a new term Network Proxy Cache. Our current understanding is that the SLIC HTTP Server Code, to improve performance even more, decided not to use the NFC to store the content from the Origin Server - but its own space we are calling the Network Proxy Cache. In the end, maybe it is not so important of a detail. But, we just wanted to be accurate.

# FRCA: Reverse Proxy Miss Scenario



MI



# Notes



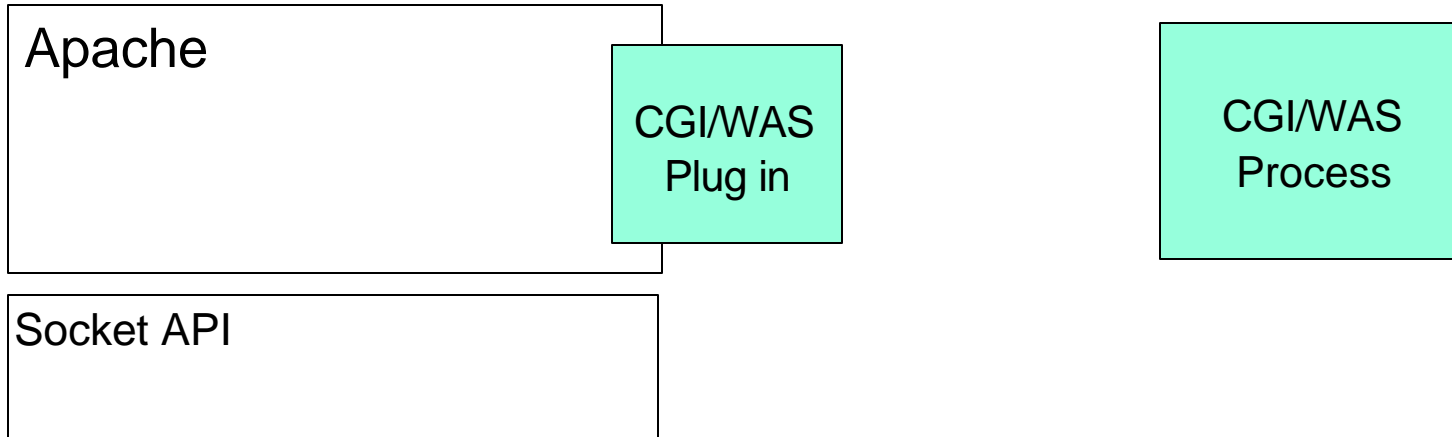
The figure above shows the FRCA reverse proxy miss scenario. That is, when FRCA recognizes that content for an incoming URI should be cached in the NFC but is not.

The steps from request through response are as follows.

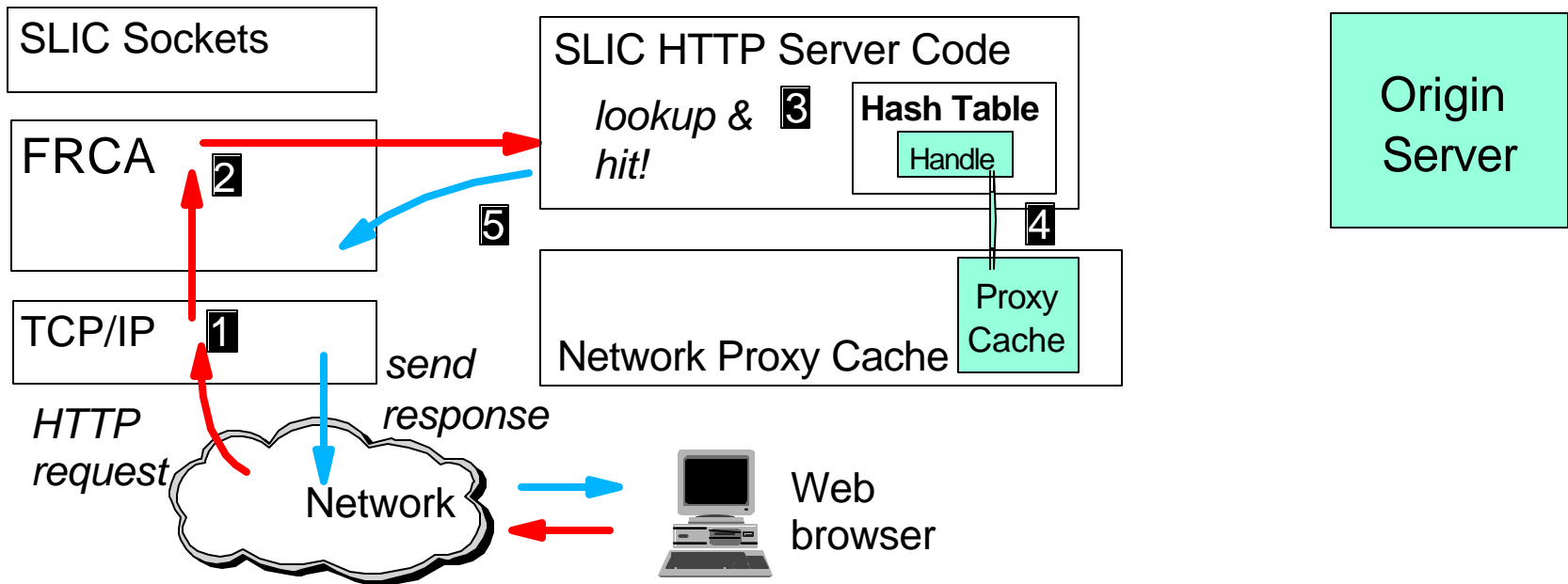
- 1 An HTTP request is received by TCP and passed to the FRCA.
- 2 FRCA sees that the incoming URI is configured for reverse proxy cache.
- 3 FRCA uses the URI as part of the handle to see if this dynamic content has been cached in the NFC. It has not (miss!).
- 4 As part of the configuration of the FRCA reverse proxy caching a new HTTP request is sent to the configured URL (for this URI). This dynamic content server (called an Origin Server) is contact via TCP/IP. This Origin Server could be located on the same iSeries server or anyplace connected via TCP/IP.
- 5 The Origin Server returns the content.
- 6 FRCA caches the content and updates the Hash Table (for the next time). The content is sent back to the web browser.

**Note:** You will notice that the NFC has been replaced in the graphics associated with FRCA reverse proxy caching with a new term Network Proxy Cache. Our current understanding is that the SLIC HTTP Server Code, to improve performance even more, decided not to use the NFC to store the content from the Origin Server - but its own space we are calling the Network Proxy Cache. In the end, maybe it is not so important of a detail. But, we just wanted to be accurate.

# FRCA: Reverse Proxy Hit Scenario



MI



# Notes



The figure above shows the FRCA reverse proxy hit scenario.

The steps from request through response are as follows.

- 1 An HTTP request is received by TCP and passed to the FRCA.
- 2 FRCA sees that the incoming URI is configured for reverse proxy cache.
- 3 FRCA uses the URI as part of the handle to see if this dynamic content has been cached in the NFC. It has (hit!).
- 4 FRCA reads the content in the NFC.
- 5 FRCA sends the content back to the web browser.

**Note:** You will notice that the NFC has been replaced in the graphics associated with FRCA reverse proxy caching with a new term Network Proxy Cache. Our current understanding is that the SLIC HTTP Server Code, to improve performance even more, decided not to use the NFC to store the content from the Origin Server - but its own space we are calling the Network Proxy Cache. In the end, maybe it is not so important of a detail. But, we just wanted to be accurate.



# FRCA: Limitations



**No SSL/TLS supported for the FRCA enabled sessions/ports**

**No authentication protection for the file in FRCA (NFC)**

- Contents should be for public access under FRCA

**No NLS code page conversion performed**

- IFS files are read in binary and loaded into the NFC cache as is

# Notes



FRCA does not support SSL and/or TLS, therefore you cannot enable FRCA cache for the sessions or ports with SSL/TLS. The reason is because SSL and TLS works above MI while FRCA works below MI.

Since you can enable FRCA cache for each listen port, the ports with SSL and without SSL can coexist in the same server and can access them as a different server using virtual host.

Once the file loaded into the NFC, it can be accessed by any users accessing files in the same server instance. Entries in the NFC are keyed by instance so there is some protection between server instances that happen to be serving the same file (the file will actually be placed in the NFC twice in this case). This is because authorization check is also performed above MI. When a request for the file that is already in the NFC comes, the file will be served without authorization check since FRCA has no way to do it. For this reason, you should enable the FRCA cache only for the contents that can be public.

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 need any code conversion for the files in the IFS to be served by the HTTP server. So this limitation should have no impacts. Even if you have the same contents in different language, they must be in different files, or if they have the same name, they must be in different directory. So, each file can be cached and served independently.

# Notes



The presents of any of the following headers in an HTTP request will force FRCA to pass the request directly to Apache 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
- wwwauthenticate
- warning

# Notes



# FRCA: Configuration: Enablement



## FRCA cache can be enabled for each separate Listen

- Listen [IP address:]port-number <optional parameter>
  - The <optional parameter> is "FRCA" and is used to enable FRCA cache
  - Examples:
    - Listen 10.5.5.5:80 FRCA
    - Listen 10.5.5.5:443

## Two directives to turn on/off other FRCA directives

- To give you the ability to turn off FRCA without having to comment out numerous local cache or reverse proxy cache directives.
- Local cache:
  - FRCAEnableFileCache On/Off
  - Enables/disables FRCA local cache for this server instance (server context)
- Reverse proxy cache:
  - FRCAEnableProxy On/Off
  - Enables/disables FRCA reverse proxy cache for this server instance (server context) and VirtualHost context

# Notes



## Directive to enable configuration and use of a specific IPaddr: port for FRCA

iSeries Apache Directive	Description	Syntax	Default	Context
Listen	To enable or disable using of the FRCA caching support for this IP Address:port	Listen IPaddr:port FRCA	off (FRCA parameter is blank)	Server Config

You can use this option on the Listen directive to enable or disable using of the FRCA caching support for this IP address and port. This directive can be used only in server config context.

### Example:

```
Listen 10.5.5.5:80 FRCA
Listen 10.5.5.5:443
```

This example enables use of FRCA cache for this server instance on port 80. Any request that comes in for port 443 (assume that port 443 is SSL/TLS traffic) is not cached by FRCA.

# Notes



## Two directives to turn on/off other FRCA directives

To give you the ability to turn off FRCA without having to comment out numerous local cache or reverse proxy cache directives you can use these two 'switch' like directives.

iSeries Apache Directive	Description	Syntax	Default	Context
FRCAEnableFileCache	To enable or disable using of the FRCA local caching support for this server instance.	on/off	off	Server Config
FRCAEnableProxy	To enable or disable using of the FRCA reverse proxy caching support for this server instance or VirtualHost context.	on/off	off	Server Config, VirtualHost

The local cache 'switch' **FRCAEnableFileCache On/Off** works only within the Server Context and will enable or disable FRCA local caching for the entire server instance. That is, If FRCAEnableFileCache is off, all other FRCA file cache related directives in the configuration file are ignored.

The default is off, so if you will be using FRCA local cache you should explicitly turn this feature on with the directive  
FRCAEnableFileCache on

# Notes



The reverse proxy cache 'switch' **FRCAEnableProxy On/Off** works within the Server Context and any VirtualHost contexts and will enable or disable FRCA reverse proxy caching. That is, If FRCAEnableProxy is off, all other FRCA reverse proxy cache related directives in the configuration file are ignored (within the context - see the notes below for an explanation of this).

The default is off, so if you will be using FRCA reverse proxy cache you should explicitly turn this feature on with the directive `FRCAEnableProxy on`

Example 1: FRCAEnableProxy on

This example enables use of FRCA proxy for the server config section for the server instance.

Example 2:

```
<virtualhost 1.2.3.4>  
FRCAEnableProxy on  
</virtualhost>
```

This example enables use of FRCA proxy for the virtual host 1.2.3.4 section for the server instance.

Notes:

- Virtual host do not inherit the FRCAEnableProxy setting from the server config.
- If FRCAEnableProxy is set to off in the server config section, only FRCA directives in server config section are ignored.
- If FRCAEnableProxy is set to off in a virtual host section, only FRCA directives in that virtual host section are ignored.



# FRCA: Configuration: Local Cache



## Limiting the size of the Local Cache and files

- `FRCACacheLocalSizeLimit nnn`
  - Use this directive to specify the maximum amount of storage, in Kilobytes, that you want to allow for FRCA local caching for this server instance.
- `FRCACacheLocalFileSizeLimit nnn`
  - Use this directive to specify the maximum file size, in bytes, that you want to allow for file caching.

## Defining when and what files are to be cached in the NFC

- `FRCACacheLocalFileStartUp ../dir/*.gif`
  - Specify file name (wild cards) to be loaded into NFC during intance startup
  - Multiple `FRCACacheLocalFileStartUp` directives can be specified
  - Does not operate recursively through sub-directories
- `FRCACacheLocalFileRunTime /dir*`
  - Similar to `FRCACacheLocalFileStartUp` but now the file(s) will not be cached in the NFC until after the first time they are requested

# Notes



## Limiting the size of the Local Cache and files

Both of these directives can be used to control the size and the number of files that are placed in the NFC by the FRCA local cache.

iSeries Apache Directive	Description	Syntax	Default	Context
FRCACacheLocalSizeLimit	Maximum storage (in kilobytes) used per server instance for caching FRCA Local Cache.	nnn Kbytes	2000 (2 Mbytes)	Server Config
FRCACacheLocalFileSizeLimit	Maximum file size (in bytes) which will be cached	nnn bytes	92160 (bytes)	Server Config

### FRCACacheLocalSizeLimit nnn

Use this directive to specify the maximum amount of storage, in Kilobytes, that you want to allow for FRCA local caching for this server instance. Example:

```
FRCACacheLocalSizeLimit 5000
```

This example caches as many files while the accumulated size is less than 5 million bytes.

#### Notes:

- The value specified here is the upper limit, the actual amount of storage allocated will be the accumulated size of the files that are cached.
- FRCACacheLocalSizeLimit can help limit your cache size when you are using wild card character to specify the files on the FRCACacheLocalFileStartUp or the FRCACacheLocalFileRunTime directives.
- If the specified size for this directive is greater than the amount of storage available in the Network File Cache (NFC), then only as many files will be cached that the NFC has space for. More information about the NFC is coming up in this presentation.

### FRCACacheLocalFileSizeLimit nnn

Use this directive to specify the maximum file size, in bytes, that you want to allow for file caching.

#### Note:

- FRCACacheLocalFileSizeLimit can help to use the cache storage for a greater number of smaller files when you are using the wild card character to specify the files on the FRCACacheLocalFileStartUp or the FRCACacheLocalFileRunTime directives.

# Notes



## Defining when and what files are to be cached in the NFC

iSeries Apache Directive	Description	Syntax	Default	Context
FRCACacheLocalFileStartUp	Path name of one or more files to cache at server start up.	path/file	None	Server Config
FRCACacheLocalFileRunTime	Directory name to dynamically cache files from during server run time based on file usage.	path/file	None	Server Config

### FRCACacheLocalFileStartUp

Example 1: FRCACacheLocalFileStartUp /www/html/index.html

- This example caches a specific file

Example 2: FRCACacheLocalFileStartUp /www/images/\*.gif

- This example caches all .gif files in the /www/images directory.

### Notes:

- Use this directive to specify the name of a file that you want to load into the NFC each time you start the server instance. You can have multiple occurrences of this directive in the configuration file.
- You can use an asterisk (\*) as a wild card character on the file names. File name matching is not recursive through sub directories. The server will only cache files in the specified directory. No files in sub directories are affected.
- If directory name begins with / then it is absolute, otherwise it is relative to the server document root
- For caching files at the server start up, only specify the path name of the files that are intended for public viewing. That is, do not specify/configure file names containing sensitive information which is not intended for general users. Files cached in the NFC are served without performing any authentication or authorization checking

### FRCACacheLocalFileRunTime

Similar to FRCACacheLocalFileStartUp but now the file(s) will not be cached in the NFC until after the first time they are requested.

# Notes



**Note:** After server start up, the first request always goes up to Apache. This is how FRCA key (URL) and the file name is correlated. You really need to get a -vv trace of Apache to see how it took two requests. The distinction between FrcaCacheLocalFileStartup and ...Runtime is when the file is actually loaded in the NFC. However, in both cases the first request is always served by Apache and the subsequent ones by FRCA, if it is cached. In -vv trace you should see the first GET has a FRCA header added to the request representing the KEY.

# FRCA: Configuration: Reverse Proxy



## Directive for FRCA reverse proxy caching

- `FRCAProxyPass /URIpath URL`
  - Specify the URI part of the dynamic HTTP request to be cached
  - Can specify the URL (protocol:host:port:URI) that serves the request
    - could be on the same iSeries or any other TCP/IP connected system

## Controlling Expiry and Refresh intervals

- `FRCAProxyCacheExpiryLimit nnn`
  - Set the HTTP expire header to this value (see notes)
- `FRCAProxyCacheRefreshInterval <path> <time>`
  - Defines how long the cached item is kept in the cache before becoming stale

## Limiting the size of the Reverse Proxy Cache and files

- `FRCAProxyCacheSizeLimit nnn`
  - Specify maximum storage for reverse proxy caching in this server instance.
- `FRCAProxyCacheEntitySizeLimit nnn`
  - Specify maximum reverse proxy response entity size that will be cached

# Notes



## Directive for FRCA reverse proxy caching

iSeries Apache Directive	Description	Syntax	Default	Context
FRCAProxyPass	This directive allows remote servers to be mapped into the space of the local server	<URI path> <URL>	None	Server, VirtualHost

Example:

Suppose the local server has address `http://some.org/`, then

```
FRCAProxyPass /mirror/foo/ http://foo.com/
```

will cause a local request for the `http://some.org/mirror/foo/bar` to be converted into a proxy request to `http://foo.com/bar`.

Once the entity is cached, the subsequent requests for the same entity will be served from the cache until the cache entity becomes stale. Any subsequent request that is not served from the cache will result in a new dynamic content to be cached. In order that multiple simultaneous requests with the same URI do not result in generating the same dynamic content multiple times, the FRCA HTTP server will provide a queuing mechanism to allow the first of multiple requests to build the cache while the others will be kept in the waiting queue. You can have multiple occurrences of this directive in the configuration file.

Notes:

- The URI part of the dynamic HTTP request(s) that you want to be cached is a required parameter. Note: A URL is made up of the protocol, a resource location (host:port) and resource id (path and object). URI is the resource ID part of the URL.
- This directive allows remote servers to be mapped into the space of the local server; the local server does not act as a proxy in the conventional sense, but appears to be a mirror of the remote server.
- FRCA Reverse Proxy does not perform authentication or authorization checking. Therefore, do not specify/configure paths or URLs that would result in responses with sensitive information which is not intended for public viewing.
- Can specify the host name and port number that serves the request. The URL could be on the same iSeries or any other connected (via intranet or Internet) system.
- If FRCAProxyPass is specified multiple times with URIs values ranging from general to more specific, then only the FRCAProxyPass directive with the most specific URI value is used.
- The value specified for the first parameter may be a static content URI, however from the SLIC HTTP server point of view it is considered dynamic since it has no way of knowing how the response is generated.
- FRCAProxyPass should only be used to specify URI for dynamically created content that is intended for public viewing. That is, do not specify URIs that results in dynamic generation of sensitive information which is not intended for general users.

# Notes



## Controlling Expiry and Refresh intervals

iSeries Apache Directive	Description	Syntax	Default	Context
FRCAProxyCacheExpiryLimit	Expiry time for FRCA proxy cacheable http documents will be set to at most nnn number of seconds into the future.	nnn	862000 (about 10 days)	Server, VirtualHost
FRCAProxyCacheRefreshInterval	The time period (in seconds) to use each FRCA proxy cached entity, for the specified path, before being re-cached.	<path> <time>	none	Server, VirtualHost

### FRCAProxyCacheExpiryLimit

Example:

```
FRCAProxyCacheExpiryLimit 3600
```

Note:

- Expiry time for FRCA reverse proxy cacheable HTTP documents will be set to at most nnn number of seconds into the future. FRCA reverse proxy cacheable HTTP documents can be at most nnn seconds out of date. If the expire header is present with the document in the response, then the lower of the two values is used.

### FRCAProxyCacheRefreshInterval

<path> is the uri part of the request.

<time> is number of seconds.

Examples:

Suppose the requested URL is `http://some.org/mirror/foo/bar` then

- `FRCAProxyCacheRefreshInterval /mirror/foo/bar 30`

will refresh any proxy cache entity having URI `/mirror/foo/bar` every 30 seconds.

- `FRCAProxyCacheRefreshInterval /mirror/foo/* 30`

will refresh any proxy cache entity starting with URI `/mirror/foo/` every 30 seconds.

Note: If the second parameter, time to live, is zero, the request will be proxied but the response content will not be cached

# Notes



## Limiting the size of the Reverse Proxy Cache and files

iSeries Apache Directive	Description	Syntax	Default	Context
FRCAProxyCacheSizeLimit	Specify maximum storage, in kilobytes, for reverse proxy caching in this server instance.	nnn (Kbytes)	2000 (2 Mbytes)	Server
FRCAProxyCacheEntitySizeLimit	Specify maximum reverse proxy response entity size, in bytes, that will be cached.	nnn (bytes)	92160 (bytes)	Server

### FRCAProxyCacheSizeLimit

Use this directive to specify the maximum amount of storage, in Kilobytes, that you want to allow for FRCA reverse proxy caching for this server instance.

Example:

- FRCAProxyCacheSizeLimit 5000

This example caches as many reverse proxy response entities while the accumulated size is less than 5 million bytes.

Notes:

- The value specified here is the upper limit, the actual amount of storage allocated will be the accumulated size of the proxy entities that are cached.
- FRCAProxyCacheSizeLimit can help limit your FRCA reverse proxy cache size.
- If the specified size for this directive is greater than the amount of storage available in the Network File Cache (NFC), then only as many files will be cached that the NFC has space for. More information about the NFC is coming up in this presentation.

### FRCAProxyCacheEntitySizeLimit

Use this directive to specify the maximum proxy response entity size, in bytes, that you want to allow for FRCA to cache.

note: FRCAProxyCacheEntitySizeLimit can help to use the FRCA cache storage for more number of average size proxy responses.

Example:

- FRCAProxyCacheEntitySizeLimit 8000

This example allows only caching of the proxy responses that are equal or less than 8000 bytes in size.



# FRCA: Configuration: Miscellaneous



## Controlling internal behavior:

- FRCACustomLog
  - FRCA can define custom log formats much like the HTTP server
  - Default is the 'normal' log style
- FRCAMaxCommTime
  - max time before sending internal logs and performance data to HTTP server
- FRCAMaxCommBufferSize
  - Sets the communication buffer size in FRCA for performance

## Controlling Specific 'uber'-dynamic Content Scenarios

- FRCACookieAware
  - Include the client cookie as part of the hash table key to the NFC
- FRCAEndofURLMarker
  - Used to cut off the client specific parameters from the URI
- FRCARandomizeResponse
  - Used to dynamically serve random files to the remote clients

# Notes



## Controlling internal behavior:

**FRCACustomLog** file-or-pipe format-or-nickname [ env=[!]environment-variable]

The FRCACustomLog directive is used to log FRCA requests to the server. A log format is specified, and the logging can optionally be made conditional on request characteristics using environment variables.

The first argument is the filename to which log records should be written. It is either a full path or relative to the current server root. If a pipe is specified, it would be the name of a program that would receive the log file information on standard in. A pipe is specified by using the pipe character "|" followed by a path to the program name (no space between them). The program name can be either a path to a QSYS program object or an IFS path to a symbolic link. The symbolic link would then link to a QSYS program. Note that data written to the pipe from the server will be in the FSCCSID that is in use by the server.

The second argument can be either a format argument or a nickname. If it is a format, it specifies a format for each line of the log file. The options available for the format are exactly the same as for the argument of the LogFormat directive. If the format includes any spaces (which it will do in almost all cases) they should be enclosed in double quotes. If the argument is a nickname, that nickname will tie back to a LogFormat directive with the same nickname specified.

The third argument is optional and allows the decision on whether or not to log a particular request to be based on the presence or absence of a particular variable in the server environment. If the specified environment variable is set for the request (or is not set, in the case of a 'env=!name' clause), then the request will be logged.

**Note:** One thing that should be mentioned is that FRCA will collect logging data down in SLIC based upon **FRCAMaxCommTime** and **FRCAMaxCommBufferSize**. When it does send the data to the HTTP server (above the MI) this data comes as a 'chunk'. The log files entries could be out-of order and might be more difficult to read. All the log data will be there - just not in the same order as it was received.

# Notes



## **FRCAMaxCommTime**

Example:

- FRCAMaxCommTime 240

Sets the maximum number of seconds to wait before the data buffer is sent from FRCA to Apache. The data being sent to Apache consists of log data, message data, and collection services data. Once the time limit has been reached, the data will be transmitted to Apache for processing.

## **FRCAMaxCommBufferSize**

Example:

- FRCAMaxCommBufferSize 4000000

Sets the communication buffer size (in bytes) in FRCA for performance. The data being sent to Apache consists of log data, message data, and collection services data. FRCA will buffer the size of data specified until the buffer is full. Once the buffer is full, the data will be transmitted to Apache for processing.

## **Controlling Specific 'uber'-dynamic Content Scenarios**

### **FRCACookieAware**

Example:

- FRCACookieAware /some\_path\_segment

This FRCA directive indicates 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 will allow content that is intended for specific individuals to be cached separately.

# Notes



## FRCAEndofURLMarker

Example:

- FRCAEndofURLMarker ###"

FRCA support can identify the end of the original URL (link) before it was modified/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 client send this request to the server it may pad the URL with some data such as: "client\_padded\_data." So the some.org server will receive the path "/some\_path/some\_parms###client\_padded\_data"

By specifying the following directive:

- "FRCAEndofURLMarker ###"

FRCA support can identify the end of the original URL (link) before it was modified/padded by the client.

## FRCARandomizeResponse <path> <string> <nnn> <mmm>

- <path> - valid paths in the form of: "/some\_path\_segment/some\_partial\_file\_nameNNN.ext" where: "NNN" marker will be replaced with a randomly generated number by FRCA before serving the response.
- <string> - The replacement string marker ("NNN") in the path.
- <nnn> - lower bound of random numbers (int)
- <mmm> - upper bound of random numbers (int)

Examples:

```
FRCARandomizeResponse /some_path/fileNNN.html NNN 1 1000
```

```
FRCARandomizeResponse /some_path/fileXXX.html XXX 200 300
```

Specify the path template, the replacement string marker, and the random number range that you would like FRCA to use to randomly select and serve files of that template.

For example, if you have 1000 "advertising" files with names: file1.html through file1000.html in your server document root, then by configuring: "FRCARandomizeResponse /document\_root\_alias\_path/fileNNN.html NNN 1 1000" and then requesting the URL: http://some\_host:port/dirpath/fileNNN.html, FRCA will randomly select and serve one of the 1000 files.

# Network File Cache: Configuration



## IPL

- The initialization of NFC occurs during IPL.
- Changes (via CHGTCPA) happen immediately

## Configuration values in CHGTCPA command:

- Enablement
- Cached file timeout
- Cache size

```
Network file cache:
Enablement . . . . . *YES
Cached file timeout . . . . . 300
Cache size . . . . . 10
                                     *DFT, *CLEAR, *SAME, *YES, *NO
                                     *NOMAX, 30-604800 sec (1week)
                                     10-100000 megabytes
```



V5R2 defaults

# Notes



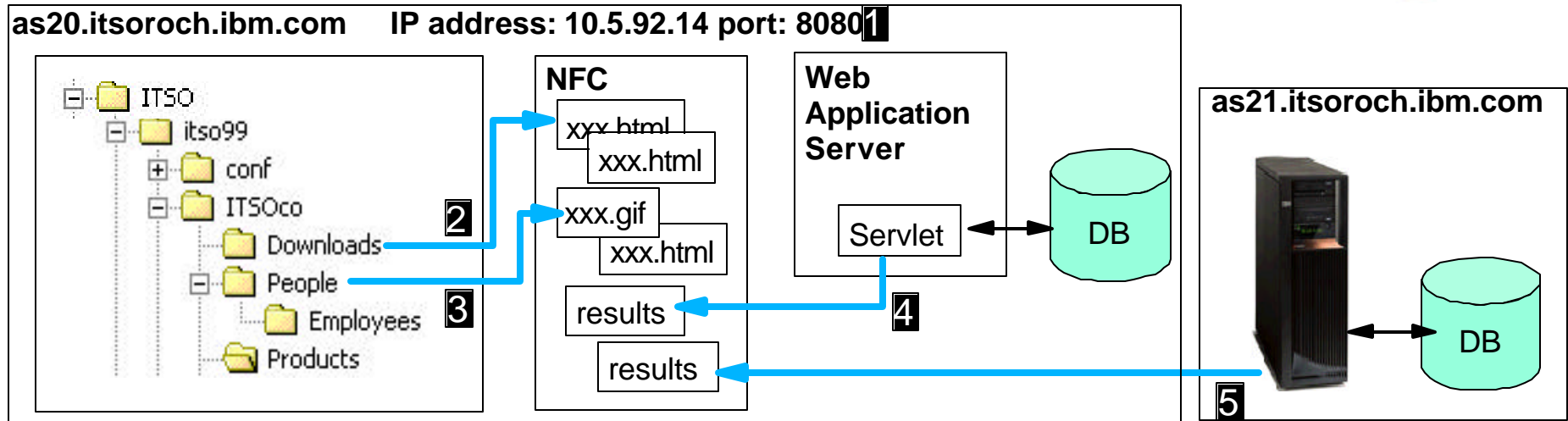
## IPL

The initialization of the Network File Cache component will occur during the IPL when the other file servers are initialized.

## Configuration values in Change TCP/IP Attributes (CHGTCPA)

- Enablement
  - Specifies whether the Network File Cache (NFC) function will be enabled on this system. The default value is \*YES.
  - When you specify \*CLEAR for this parameter, it immediately clear the entire Network File Cache. After the cache is cleared, the previous Network File Cache values will be retained.
- Cached file timeout
  - Specifies the maximum amount of time, in seconds, that a file can be cached in the Network File Cache. This ensures that a file is refreshed at a regular interval. A value of \*NOMAX is available.
  - A cache time can be specified when NFC is not enabled.
- Cache size
  - Specifies the maximum amount of storage that may be used by the NFC for the entire system. This is the accumulative storage used by all TCP servers for loading files.
  - A cache size can be specified when NFC is not enabled.

# FRCA: Configuration Examples



Configuration file  
HTTP server: ITSO99

```

...
2 Listen 10.5.92.14:8080 FRCA 1
...
5 FRCAEnableFileCache On
...
11 FRCACacheLocalFileStartup /ITSO/itso99/ITSOco/Downloads/*.htm 2
12 FRCACacheLocalFileRuntime /ITSO/itso99/ITSOco/People 3
...
15 FRCAEnableProxy On
16 FRCAProxyPass /servlet/ http://10.5.92.14:8080/servlet/ 4
17 FRCAProxyCacheRefreshInterval /servlet/ 300
...
5 20 FRCAProxyPass /cgi-bin/ http://as21.itsoroch.ibm.com:9999/cgi-bin/
21 FRCAProxyCacheRefreshInterval /cgi-bin/ 180
    
```

# Notes



Here are descriptions of this example.

## **Listen 10.5.92.14:8080 FRCA**

Specifying Listen directive with the parameter "FRCA" enables FRCA cache for this port.

### **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.

### **FRCACacheLocalFileStartUp /ITSO/itso99/ITSOco/Downloads/\*.html**

By specifying this directive, the files that have .html extension in the directory /ITSO/itso99/ITSOco /Download are all cached when you start the server ITS099.

### **FRCACacheLocalFileRunTime /ITSO/itso99/ITSOco/People/\***

This directive makes all files in the directory /ITSO/itso99/ITSOco/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.

### **FRCAEnableProxy On**

This directive enables FRCA proxy.

### **FRCAProxyPass /servlet/ http://10.5.92.14:8080/servlet/**

In this example, specifying /servlet in URI causes to run a servlet on the application server. By specifying the directive FRCAProxyPass like this example, the result of the servlet can be cached in the NFC for certain period, that is specified by the directive `frcaproxycacherefreshinterval`.

**Note:** In this directive of the example, the target URL has the same IP address and port as the ones this server listens.

In this case, FRCA should understand that this URL is of the same server, and passes the request to the correct route without any looping problem.

### **FRCAProxyCacheRefreshInterval /servlet/ 300**

As described above, this directive specifies the interval of refreshing cached data of FRCA proxy.

### **FRCAProxyPass /cgi-bin/ http://as21.itsoroch.ibm.com:9999 /cgi-bin/**

By specifying this directive, the request for CGI program is rerouted to the target host `as21.itsoroch.ibm.com:9999`, that is different iSeries, and the result is cached in the NFC in the source system.



# Notes



```
Listen 3355
Listen 4455 FRCA
#*** frca local cache
frcaenablefilecache on
frcacachelocalsizelimit 4000
frcacachelocalfilesizelimit 10000
frcacachelocalfilestartup /tpcw/frcas*
frcacachelocalfileruntime /tpcw/frcar*
#*** frca proxy
#frcaenableproxy on
#frcaproxy cachesizelimit 5000
#frcaproxy cacheentitysizelimit 20000
#frcaproxy pass /frca_proxg/ http://lpar147m:6655/
#frcaproxy cacherefreshinterval /frca_proxg/ 120
#*** frca advance
#frcaendofurlmarker zzzz
#frcarandomizeresponse /frca_proxy/randomNNN NNN 1 100
#frcacookieaware /frca_proxg/
#*** frca data pipe
#frcamaxcommbuffersize 10000
#frcamaxcommtime 20
AsyncIO On
Addtype image/gif GIF
Addtype image/jpg JPG
AliasMatch ^(.*)\.htm$ /tpcw/$1.htm
AliasMatch ^(.*)\.html$ /tpcw/$1.html
ErrorLog logs/frca_err.log
LogLevel debug
```

```
(continued)
HostNameLookups off
ThreadsPerChild 4
RuleCaseSense off
#IconPath /QIBM/HTTPSVR/Icons/
MaxCGIJobs 1000
CGIMultiThreaded on
#MaxThreadedCGIJobs 100
CGIRecyclePersist off
LogFormat "%h %t \"%r\" %s %b" "TPC-W Required"
CustomLog logs/frca_acc.log "TPC-W Required"
#FRCACustomLog logs/frca_facc.log "TPC-W Required"
<Directory />
  Options None
  AllowOverride None
</Directory>
UseCanonicalName Off
KeepAlive ON
KeepAliveTimeout 600
#TimeOut 300
TimeOut 600
MaxKeepAliveRequests 9999
Options IncludesNOEXEC Indexes
ListenBacklog 5000
HotBackup Off
IdentityCheck off
LogTime LocalTime
```

# Notes



# FRCA: Collection Service



## How to know the files are cached

- Communication trace
  - This tells us where the files are being served from

## How to know the proxy cache works

- WRKACTJOB option 12 for Work with Thread
  - You can see if the threads using CPU
  - For servlet or JSP, you have to find the thread first
  - Collection Services statistics

## FRCA logs

## Access logs

# Notes



## How to know the files are cached

You can see if the files are served by FRCA or not in the communication trace. When the files are served by FRCA, the trace shows like:

```
HTTP/1.1 200 OK..DATE: MON, 25 MAR 2002 22:24:43 GMT..SERVER: APACHE/2.0.32(FRCA)..ACCEPT-RANGES:
BYTES..CONNECTION: CLOSE..LAST-MODIFIED: MON, 25 MAR 2002 22:14:31 GMT..CONTENT-TYPE: TEXT/HT ....
```

This is a downstream data comes from iSeries.

If the file is not served from FRCA cache, the trace shows like:

```
HTTP/1.1 200 OK..DATE: MON, 25 MAR 2002 22:25:13 GMT..SERVER: APACHE..LAST-MODIFIED: MON, 25 MAR 2002
22:04:13 GMT..ETAG: "BEC2-321-9DCFB140"..ACCEPT-RANGES: BYTES..CONTENT-LENGTH:801..KEEP-ALIVE:
TIMEOUT=15,MAX=100..CONNECTION: KEEP-ALIVE..CONTENT-TYPE: TEXT/HTML; CHARSET=WINDOWS-1252
```

## How to know the proxy cache works

When you use FRCA proxy cache to serve servlets, CGI or other dynamic contents, the way to make sure if the cache is working is to watch the CPU utilization of the job or threads.

But you have to find which thread you should watch. This can be difficult when multiple threads are active at the same time to serve multiple requests of CGI or servlet.

## FRCA logs

### Access logs

# New V5R2 HTTP data collected



## Collection Services

- Analyze HTTP server activity and transaction types
- Types of information
  - Divided into categories
    - Examples: Server (static Web pages), CGI, WebSphere, Tomcat, Proxy, FRCA, SSL
  - Number of requests, responses, error responses
  - Processing time
  - Responses served from a cache
  - Number of bytes sent and received

## Performance Tools Reports

- Information on the transactions processed by HTTP Server jobs

# Notes



Web-based transaction processing and web-serving environments continue to grow in importance. We are enhancing the performance of these environments by providing improvements to SSL, implementing Fast Response Cache Accelerator (FRCA), and continued work with asynchronous I/O.

## **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).

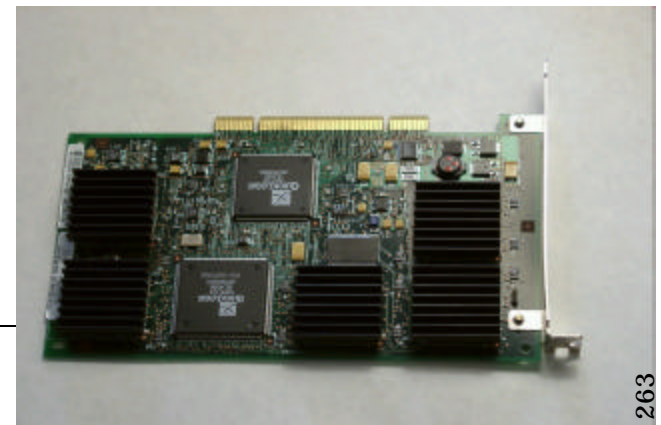
**Performance Tools for iSeries** has been enhanced to generate reports based on the HTTP performance data. The reports contain information on the transactions processed by HTTP Server jobs.

Refer to the 2002 ITSO Forum: Performance presentation for an overview of the report enhancements.

# Cryptographic Accelerator - 2058-001



- Offloads selected portions of cryptographic processing from host processors
  - CRTDEVCRP DEVD(CRYACL) RSRRCNAME(CMN002) PKAKEYFILE(\*NONE) DESKEYFILE(\*NONE) TEXT('2058 (iSeries Feat #4805)')
- iSeries SSL uses for SSL session "handshake" processing (just as SSL can do for older 4758 (#4801/#4802 Cryptographic Coprocessor))
- Single Cryptographic Accelerator can support up to 1000 full SSL handshakes per second
  - Up to 4 Cryptographic Accelerators supported per system
- V5R2 performance metric:
  - Collection Services counts SSL and non-SSL connections
  - Performance Tools for iSeries System Report shows connection counts



# Notes



This new Cryptographic Accelerator "card" is used by the system on an SSL connection handshake process (negotiates the level of SSL support and key exchange information to be used by an SSL session). You still use a digital certificate to perform the SSL processing and define the application (HTTP serving, Management Central, Telnet, and so forth) to use SSL or not.

The older 4758 technology and the new technology 2058 are used for SSL handshaking and not for the actual data transmission encryption. This is because the overhead of constantly passing data to the hardware to be encrypted on output and decrypted upon input negate the benefits of having the hardware perform the function.

The V5R1 and V5R2 Performance Capabilities Reference manual has test results showing the handshaking performance improvements for both the older 4758 technology and the new technology 2058. Note that the 4758 provides a broader range of encryption algorithms and financial industry encryption capabilities that are not supported with the 2058.

The detailed Security presentation contains additional information. See V5R2 Information Center -> Security for the most complete coverage of 4758 and 2058 capabilities.





# Search Engine

# Notes



On the iSeries the search engine comes in two logical pieces that are related to each other. They are “iSeries Webserver Search Engine” and “iSeries Webserver Search Engine Web Crawler” .

In this part , we will briefly describe how these functions are implemented on the HTTP Server (powered by Apache).

- ▶ Search Engine and Web Crawler

# Search Engine and Web Crawler



## iSeries Webserver Search Engine

- Makes your system to be a searchable site
  - ▶ Collect all documents into a single directory
  - ▶ Create a search index
  - ▶ Document list is created that contains the indexed documents
  - ▶ Customize your search forms with supplied HTML section
  - ▶ Set up your HTTP server correctly for the search forms
  - ▶ Keep your index up to date

## iSeries Webserver Search Engine Web crawler

- Program that crawls the URL you provide and download the Web page
- Builds a document list using downloaded Web pages
- The document list can be used to create a search index

# Notes



## iSeries Webserver Search Engine

If you want to allow others to search through documents on your server, you will need to set up your system to be a searchable site. Doing this is very easy with the new iSeries Webserver Search Engine. There are just a few administrative tasks you need to do.

These tasks can be summarized as follows:

- ▶ First, collect all of the related documents into a single directory on your iSeries. You may use either the Root (/) directory of the IFS or the QSYS.LIB file system. Using the IFS system allows you to easily port your files from a PC onto the iSeries.
- ▶ Next you will need to create a search index. An index is the collection of all of the selected documents in your directory. They are stored in a special indexed form. In the indexing process, the search engine takes each document provided in a document list, 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 could be done on documents that are not indexed.
- ▶ 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.
- ▶ Once you have created the search index, you can test it from the search administration form. This will allow you to see all of the different options available to select for a search, such as fuzzy or precise.
- ▶ Now you are ready to set up the Webserver Search Engine to run on your Web site. A short HTML section has been supplied that can be added to your web page as well as a Net.Data macro containing all of the HTML you will need. This allows you to customize your search and search results forms. You may just 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.
- ▶ Once you have decided how you want to present your search forms, you will need to make sure the HTTP server you use contains the correct directives in the configuration to run the Net.Data macro and to make sure 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).
- ▶ When all of this is completed, you are ready to do some searches!
- ▶ It is important to keep your index up to date. If you modify your documents from time to time, you want to make sure your users are finding the most current information. We have supplied a way for you to update your index. You can use the same document list you used when you originally created your index. We will index any changed files that were previously indexed. You can also add a new set of documents to an index that already exists as well as delete some of the documents from your index. This is just 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>

# Notes



## **iSeries Webserver Search Engine Web Crawler**

The 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 you do when you enter a URL on your browser and then click on various links to go to new Web pages.

The crawling program starts by finding the URL 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.

The Web crawler extends the capability for building a document list. As each file is downloaded, the local path plus the original URL is added to your document list. This document list can then be used to create a search index. Search results for this type of index will 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 will be taken to the actual page that was found during crawling.

When you select 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 will take 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 have specified.

The 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 lots 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. Additionally 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>

# Notes





# Apache Modules

# Notes



This part introduce you the concept of Apache modules that can be used for extending the functionality of the HTTP server (powered for Apache). And also shows you the concept of Apache Portable Runtime (APR).

- ▶ Apache module design overview
- ▶ Apache Portable Runtime (APR)



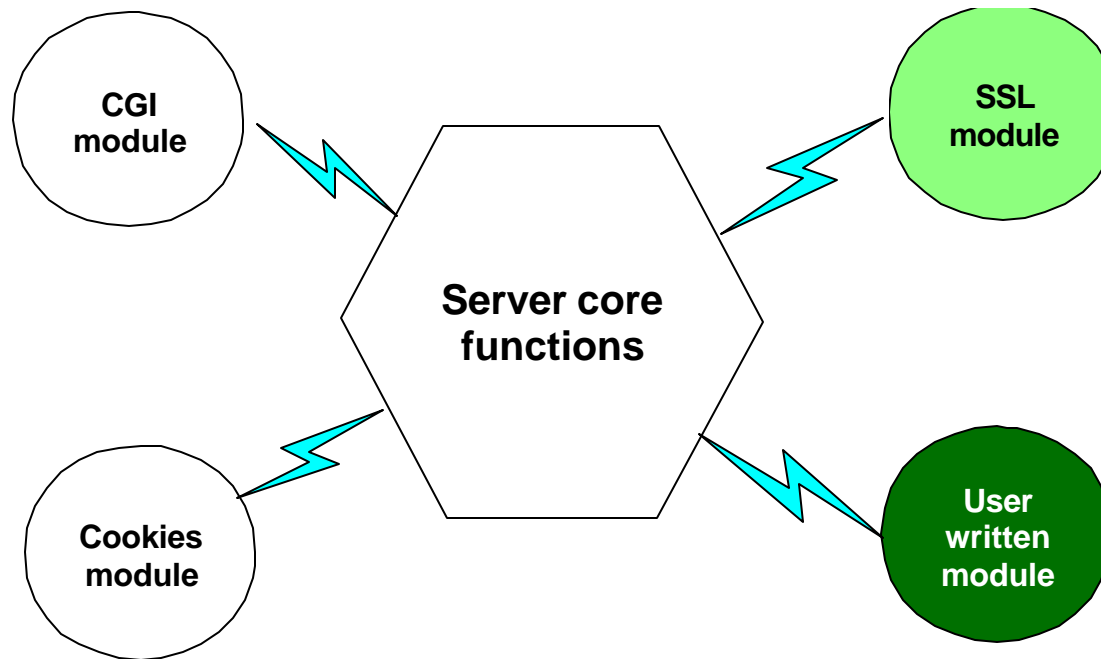
# Apache Module Design Overview



**Modules extend the Apache HTTP server's core function**

**Used only when loaded with the LoadModule directive**

**You can write your own module to extend the function**



# Notes



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

- Window's Dynamic Link Libraries (DLL)
- UNIX's shared object libraries
- OS/400's 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 very 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 need 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 v2.0 supports about 137 directives. Of those, 53 are in the core functions. The remainder are in separate modules that have been compiled into the code. The LoadModule directive must be used to activate the directives in these modules.

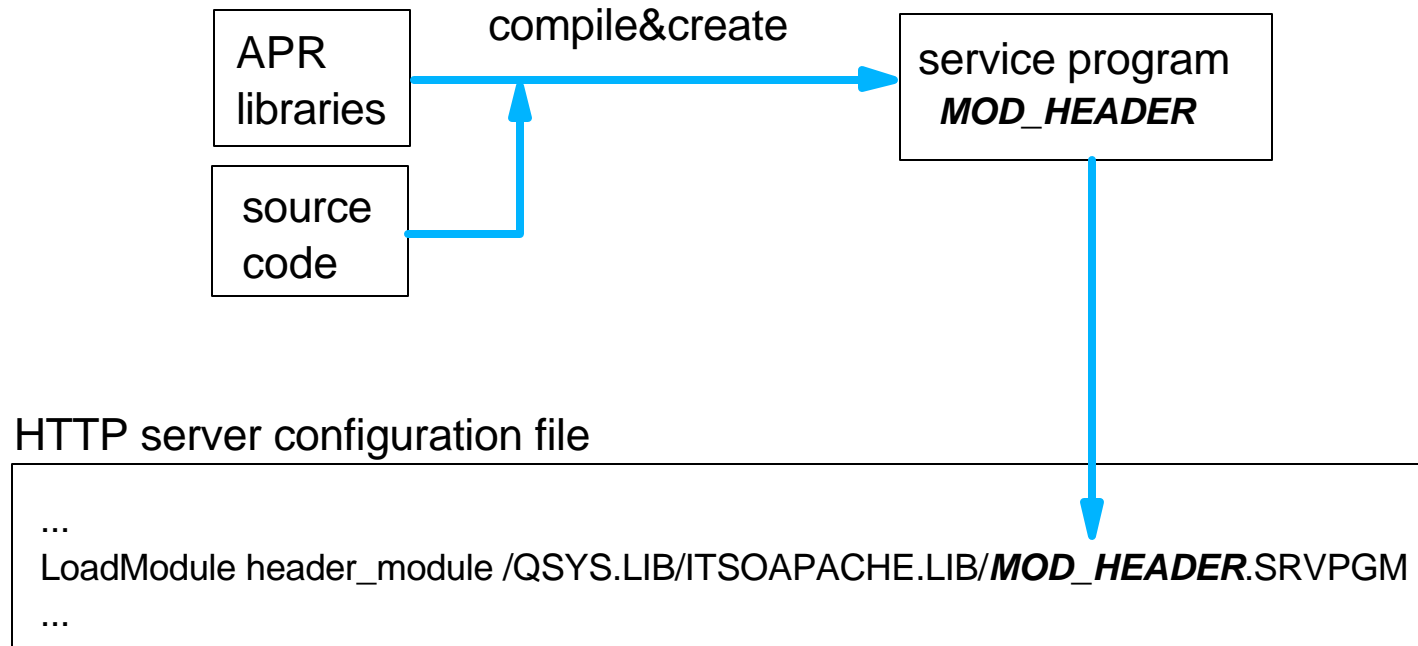
As shown in the figure above, the Apache HTTP server's core functions are extended with a variety of different modules. In some cases (CGI module and a Cookie module) the modules are 'built in' extensions to the server's core functions and do not need an explicit LoadModule to use. In other cases (for 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.

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 very popular HTTP server.

# Apache Portable Runtime (APR)



- APR provides a set of routines to write your own modules
- The modules can extend the function of Apache
- The source code can be cross-platform



# Notes

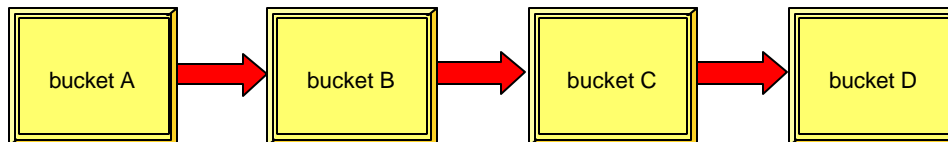


The HTTP Server (powered by Apache) can extend its functionality in specific areas of your server using modules. For example, a module could be configured to create a new type of authentication that is not available with the shipped HTTP Server (powered by Apache). Before the module can be used by your HTTP Server (powered by Apache), it must be compiled and saved in the QSYS directory. In addition, the LoadModule directive must be entered in your server configuration file along with any specific context required information.

Using your own module, the data that was generated by an earlier module can be modified. This concept is called buckets and brigades as seen in the figure below. 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.
- 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 here is linked list of buckets.



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 current 1.3 version of the Apache server. One of the big differences 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 you should first update it to the new version 2.0 APR module. Then, the port to the 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.



# Problem Determination

# Notes



The HTTP Server (powered by Apache) provides several tools that help you to determine the problems which may occur in any part of your server. This part shows you the tools that you can use for problem determination, and gives some descriptions how to use them.

- ▶ Working with configuration files
- ▶ Joblogs
- ▶ Server logs: Access logs
- ▶ Server logs: Error logs & script logs
- ▶ HTTP server traces
- ▶ Other startup parameters

# Working with Configuration Files



- Creating by GUI is mandatory
  - Changing by GUI is recommended
  - GUI can highlight the error made by manual alteration (see below)
- For Apache 'gurus' you can edit the directives directly
  - Edit Configuration File (from GUI)
    - Allows copy and paste
  - Edit File (EDTF) from green screen
  - Apache configuration files are created as unicode
    - copy and paste from GUI to PC 'wordpad' or other editor. Map the network drive...



```
# Configuration originally created by Apache Setup
Listen 9.5.92.28:80
DocumentRoot /www/brsmith04/htdocs
ServerRoot /www/brsmith04
BogusDirective /not/right/not/never
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerM
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i
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
```

**Display Configuration File**

HTTP server: BRSMITH04  
Selected file: /www/brsmith04/conf/httpd.conf

```
1 # Configuration originally created by Apache Setup Wizard Tue Jul 02
2 Listen 9.5.92.28:80
3 DocumentRoot /www/brsmith04/htdocs
4 ServerRoot /www/brsmith04
5 BogusDirective /not/right/not/never
   Keyword is not valid
6 Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch +Inclu
```

# Notes



Manually editing the configuration file requires care, patience, knowledge of the configuration directives and finally a good backup of the original file. Manually editing your httpd.conf is not recommended unless you really know what you are doing and you have a solid experience with the Apache configuration directives.

The recommended way of changing or creating your HTTP Server (powered by Apache) configuration is using the Graphical User interface, or GUI. The GUI also sports the best tools for displaying and editing configuration files.

From the main Configuration panel select the Display Configuration File option. This will bring up the content of your configuration file just like the server sees it. This is really important if you have 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, but it won'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.

For Apache 'gurus' you can edit the directives directly

- Edit Configuration File (from GUI)
  - Allows copy and paste. Usefull once you have a <directory> or <VirtualHost>, for example, configuration all setup and tested. Then you can just copy and paste the configuration and then change a few of the parameters rather than having to go through the GUI multiple times...
- Edit File (EDTF) from green screen
- Apache configuration files are created as unicode
  - copy and paste from GUI to PC 'wordpad' or other editor. Map the network drive...



# Joblogs



- The first place to look for information
- "Message logging" settings in the JOB D QHTTPSVR/QZHBHTTP
- Produced under the user profile QTMHHTTP by default
  - The user can be changed using ServerUserID directive

Server Properties

- General Server Configuration
- Container Management
- Virtual Hosts
- URL Mapping

## General Server Configuration ?

General Settings | Welcome Pages | Configuration Includes | Advanced

Server user profile:  ?

Server CCSID:  or...  ?

Client CCSID:  ?

Include server description:  ?

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

# Notes



HTTP server joblogs 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 (JOBDD) in use. The JOBDD used by the HTTP Server (powered by Apache) is QZHBHTTP in the QHTTSPVR library. Changing its message logging settings will always influence the content of your server joblogs.

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 (CHGJOBDD) CL command for usage information. Also remember that \*SECLVL will generate a highly verbose joblog for every server job, and is therefore not to be chosen as a default setting.

Joblogs will always be produced under the default QTMHHTTP profile unless you choose to use a different one adding a ServerUserID directive in your configuration file. The figure above shows you how you can change the default user using the GUI.

Messages in your server joblogs will often contain helpful hints for problem determination, like:

- the name of a failing module
- illegal configuration options
- usage of a deprecated directive
- the number of the line where an error was found

In order 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 in the GUI.

```
HTP8006 Diagnostic      40 10/12/01 10:23:25 QZSRAPR   QHTTSPVR *STMT QZSRCORE QHTTSPVR
      From module ..... : QZSRSNM
      From procedure ..... : sendMessageToJobLog
      Statement ..... : 11
      To module ..... : HTTP_CONFI
      To procedure ..... : ap_walk_config_sub
      Statement ..... : 9
      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 10/12/01 10:23:25 QZSRAPR   QHTTSPVR *STMT QZHBHTTP QHTTSPVR
      From module ..... : QZSRSNM
      From procedure ..... : sendEscapeWithMessageFile
      Statement ..... : 2
      To module ..... : HTDAEMON
      To procedure ..... : BigSwitch_FiPPc
      Statement ..... : 1066
      Message ..... : HTTP Server Instance ITSOSRV1 failed during startup
```

# Server jobs



**First job is "parent" process**

**Then come jobs for logging**

- One per log file
  - Always an error log
  - possibly others... including access log

**Then comes the primary "child" process**

**Followed by the hot backup "child" process**

**Followed by processes for CGI requests and/or non-threadsafe file system access**

Opt	Subsystem/Job	User	Type	CPU %	Function	Status
—	BRSMITH04	QTMHHTTP	BCH	.0	PGM-QZHBHTTP	SIGW
—	BRSMITH04	QTMHHTTP	BCI	.0	PGM-QZSRLOG	SIGW
—	BRSMITH04	QTMHHTTP	BCI	.0	PGM-QZSRLOG	SIGW
—	BRSMITH04	QTMHHTTP	BCI	.0	PGM-QZSRHTTP	SIGW
—	BRSMITH04	QTMHHTTP	BCI	.0	PGM-QZSRHTTP	SIGW

# Notes



Here is an example with a simple server BRSMITH04. Here is a portion of the Work with Active Jobs (WRKACTJOB) screen:

Opt	Subsystem/Job	User	Type	CPU %	Function	Status
	ADMIN	QTMHHTTP	BCH	.0	PGM-QZHBHTTP	SIGW
	ADMIN	QTMHHTTP	BCI	.0	PGM-QZSRLOG	SIGW
	ADMIN	QTMHHTTP	BCI	.6	PGM-QZSRHTTP	SIGW
	ADMIN	QTMHHTTP	BCI	.0	PGM-QYUNLANG	TIMW
	BRSMITH04	QTMHHTTP	BCH	.0	PGM-QZHBHTTP	SIGW
	BRSMITH04	QTMHHTTP	BCI	.0	PGM-QZSRLOG	SIGW
	BRSMITH04	QTMHHTTP	BCI	.0	PGM-QZSRLOG	SIGW
	BRSMITH04	QTMHHTTP	BCI	.0	PGM-QZSRHTTP	SIGW
	BRSMITH04	QTMHHTTP	BCI	.0	PGM-QZSRHTTP	SIGW

You can take a look at the joblog for each of these jobs. At the start of the joblog the job will tell you what kind of Apache job this is:

- 1st: This is the manager job for HTTP Server instance BRSMITH04. That is, the "parent" process.
- 2nd: This is a logging job for HTTP Server BRSMITH04. Usually Error log (which is always present)
- 3rd: This is a logging job for HTTP Server BRSMITH04. Second log is usually Access log (if configured)
- 4th: This is the primary job for HTTP Server instance BRSMITH04. That is, the primary "child" process.
- 5th: This one does not tell you, but it is the hot backup "child" process.

# Server Logs: Access Logs



- Powerful tool for problem determination and performance tuning
- Record every single request received by the server

```
10.1.62.20 -- [09/Nov/2001:14:04:33 -0600] "GET /Services_Np1.gif HTTP/1.1" 200 1753 "http://10.1.92.28:8002/" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:33 -0600] "GET /Projects_Np1.gif HTTP/1.1" 200 1753 "http://10.1.92.28:8002/" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:33 -0600] "GET /Downloads_Np1.gif HTTP/1.1" 200 1773 "http://10.1.92.28:8002/" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:33 -0600] "GET /SiteMap_Np1.gif HTTP/1.1" 200 1763 "http://10.1.92.28:8002/" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:33 -0600] "GET /BuiltByNOF.gif HTTP/1.1" 200 1641 "http://10.1.92.28:8002/" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:33 -0600] "GET /Home_NBanner.GIF HTTP/1.1" 200 2089 "http://10.1.92.28:8002/" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:33 -0600] "GET /Ss02043.JPG HTTP/1.1" 200 21705 "http://10.1.92.28:8002/" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:36 -0600] "GET /Products/products.html HTTP/1.1" 200 6643 "http://10.1.92.28:8002/" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:36 -0600] "GET /Home_Np1.gif HTTP/1.1" 200 1729 "http://10.1.92.28:8002/Products/products.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:36 -0600] "GET /Products_Hp3.gif HTTP/1.1" 200 1829 "http://10.1.92.28:8002/Products/products.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:36 -0600] "GET /Products_NBanner.GIF HTTP/1.1" 200 2219 "http://10.1.92.28:8002/Products/products.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:37 -0600] "GET /SiteMap/sitemap.html HTTP/1.1" 200 10674 "http://10.1.92.28:8002/Products/products.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:37 -0600] "GET /SiteMap_Hp3.gif HTTP/1.1" 200 1835 "http://10.1.92.28:8002/SiteMap/sitemap.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:37 -0600] "GET /SiteMap_NBanner.GIF HTTP/1.1" 200 2265 "http://10.1.92.28:8002/SiteMap/sitemap.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:37 -0600] "GET /SiteMap/a_Sitemap_Image.gif HTTP/1.1" 200 6761 "http://10.1.92.28:8002/SiteMap/sitemap.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:37 -0600] "GET /a_SatelliteDataIcon_4.gif HTTP/1.1" 200 1632 "http://10.1.92.28:8002/SiteMap/sitemap.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:38 -0600] "GET /Services/services.html HTTP/1.1" 200 14526 "http://10.1.92.28:8002/SiteMap/sitemap.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:38 -0600] "GET /Services_Hp3.gif HTTP/1.1" 200 1825 "http://10.1.92.28:8002/Services/services.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:38 -0600] "GET /Calendar_Ns1.gif HTTP/1.1" 200 1802 "http://10.1.92.28:8002/Services/services.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:38 -0600] "GET /Commuter_Ns1.gif HTTP/1.1" 200 1807 "http://10.1.92.28:8002/Services/services.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:38 -0600] "GET /Contacts_Ns1.gif HTTP/1.1" 200 1801 "http://10.1.92.28:8002/Services/services.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:39 -0600] "GET /Forms_Ns1.gif HTTP/1.1" 200 1787 "http://10.1.92.28:8002/Services/services.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:39 -0600] "GET /Policies_Ns1.gif HTTP/1.1" 200 1794 "http://10.1.92.28:8002/Services/services.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:39 -0600] "GET /Services_NBanner.GIF HTTP/1.1" 200 2243 "http://10.1.92.28:8002/Services/services.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:39 -0600] "GET /Postings_Ns1.gif HTTP/1.1" 200 1808 "http://10.1.92.28:8002/Services/services.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
10.1.62.20 -- [09/Nov/2001:14:04:39 -0600] "GET /Services/a_ArrowLine_6.gif HTTP/1.1" 200 1615 "http://10.1.92.28:8002/Services/services.html" "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
```

F03EP01v5r2ip6AqPBZ

# Notes



Server logs are most useful for monitoring server activity and keeping track of user access, and a valid aid in debugging as well. By carefully examining these logs we can discover the reason behind the most common error messages and eventually point out configuration errors.

## Access logs

Access logs can be an extremely powerful tool for both problem determination and performance tuning. They record every single request received by the server and can also be customised through the creation of custom log formats. Look at the figure on the right, showing a sample access log configuration. Each one of our log files is based on a different format. For now, let's just notice that our access log will use the combined format.

Let's now see what kind of information will be stored in these three logs basing on the log format we have chosen. The following figure shows us four user-defined formats and the type of information that we want them to collect. Each one of the case sensitive tokens we can use in a format definition represents a different piece of information about the client, the request received or the status of client-server communications.

Logging ?

Error Logs Script Logs FRCA Logs

User Tracking (Cookies) Custom Environment Variables

General Settings Custom Formats Custom Logs

Log formats: ?

	Format name	Log format
Example	MyFavorite	%h %a
<input checked="" type="radio"/>	combined	%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"
<input type="radio"/>	cookie	%{Cookie}n \"%r\" %t
<input type="radio"/>	agent	%{User-agent}i
<input type="radio"/>	referer	%{Referer}i -> %U
<input type="radio"/>	common	%h %l %u %t \"%r\" %>s %b

Add

# Notes



We will now analyze an access log entry, looking for the data we included in the associated log format. Let's say we want to access our server's sample homepage. Let's open a browser window and just type in `http://servername:port` in the address bar. The last line in our server's access log will now look like the one in this figure.

```
Browse : /qibmwww/itsosrv1/logs/access_log
Record :   1   of   2 by 18           Column :   1   130 by 131
Control :

....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+....0....
10.5.100.130 unknown - [14/Oct/2001:16:53:40 +0900] "GET / HTTP/1.0" 200 959 "-" "Mozilla/4.76 [en] (Win
10.5.100.58 unknown - [15/Oct/2001:10:48:47 +0900] "GET / HTTP/1.1" 200 959 "-" "Mozilla/4.0 (compatible
*****End of Data*****
```

For more information on customizing log formats see the log format article in the HTTP Server documentation center ([http://publib.boulder.ibm.com/pubs/html/iseres\\_http/v5r1/info/rzaie/rzaielogformat.htm](http://publib.boulder.ibm.com/pubs/html/iseres_http/v5r1/info/rzaie/rzaielogformat.htm)) and the Worldwide Web Consortium (W3C) logfile standards at <http://www.w3.org>.

# Notes





# Server Logs: Error Logs & Script Logs



## Error Logs

- Resides in the /logs subdirectory of the server root by default
- Useful for debugging configuration problem
- Configuration change, end/start, system errors are recorded
- Logging levels

## Script Logs

- All CGI parsed data is recorded
- Impacts on CGI performance

The screenshot shows a configuration window with three tabs: "Error Logs", "Script Logs", and "FRCA Logs". The "Error Logs" tab is active. The configuration includes:

- "Enable error logging:" set to "Enabled" with a help icon.
- "Error log:" set to "logs/error\_log" with a help icon.
- "Expiration:" set to "0" with a "Days" dropdown and an "or..." dropdown with a help icon.
- "Maximum cumulative size:" set to "0" with a "Bytes" dropdown and an "or..." dropdown with a help icon.
- "Error log entries:" with a help icon.
- "Logging level:" set to "Warning" with a dropdown menu.
- A list of logging levels with checkboxes: Emergency, Alert, Critical, Error, Warning, Notice, Informational, and Debug. The checkboxes for Emergency, Alert, Critical, Error, and Warning are checked.

# Notes



## Error log

Servers created using the GUI wizard will 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, like when a document is not accessible or the URI (Universal Resource Identifier, that is the path you add after the server address) you're pointing to is not working as expected. Error logs will also keep track of configuration changes, server end/restart and record some system errors. Be aware that any problem detected after server initialization will most likely not be recorded in the server joblogs (unless a critical condition occurs), but in the error log.

Apache now requires that you have an error log running. The Create New HTTP Server wizard will not ask you if you want to create a combined (access and error) log (V5R1) but will ask if you want to create an access log. This assumes that you must have the error log.

**Logging level:** adjusts the complexity for messages recorded in the error logs. The drop-down menu provides all valid error logging options. When a particular level is specified, messages from all other levels of higher significance will be reported as well. For example, when Critical is specified, then messages with log levels of Alert and Emergency will also be posted. Using a level of at least Critical is recommended. The default level is Warning. This field is optional. Directive: LogLevel

The information located directly underneath the drop-down menu displays which logging options are currently active:

- If Emergency, system is unusable messages ("Child cannot open lock file. Exiting.").
- If Alert, action must be taken immediately messages ("getpwuid: couldn't determine user name from uid.").
- If Critical, critical conditions messages ("Socket: Failed to get socket, exiting child.").
- If Error, error conditions messages ("Premature end of script headers.").
- If Warning, warning conditions messages ("Child process 1234 did not exit, sending another SIGHUP.").
- If Notice, normal but significant conditions messages ("httpd: caught SIGBUS, attempting to dump core in...").
- If Informational, informational messages ("Server seems busy, (you may need to increase StartServers or Min/MaxSpareServers)...").
- If Debug, debug-level messages ("Opening config file...").

## Script log

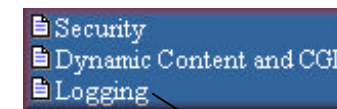
Script logs record all CGI parsed data, and can therefore have a significant impact on CGI performance. They should be used for debug purposes only and not be kept active all the time. Being a mere debug tool they are not customizable, save for the maximum amount of data to be collected.

# V5R2 Logging Enhancements



## Log rollover

- Configure log files to be closed and new log files opened
- Hourly, Daily (default), Weekly, Monthly

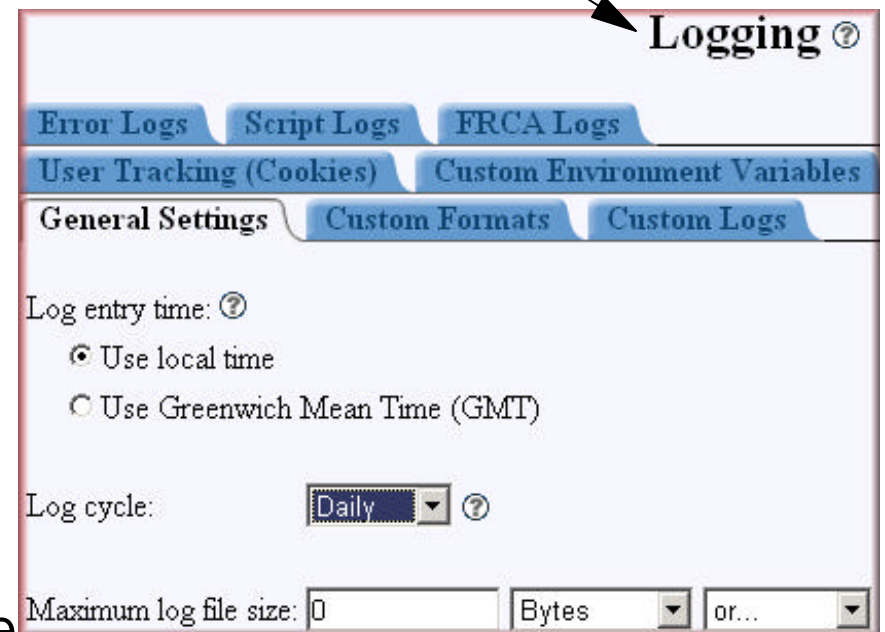


## Log maintenance (archival)

- Configure directory where the server does log file maintenance
- Can maintain by 2 categories
  - Age
  - Size

## Maximum log size

- Server stops logging when the log file size reaches the maximum



## Logging to QSYS source physical files

# Notes



## General Settings

The Logging General Settings tab 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.

**Log entry time:** allows you to select which time format each log entry time stamp will follow. The default value is local time. Directive: LogTime

- Inherit: indicates that the time format each log entry time stamp is inherited from a higher level context.
- Use local time: each log entry is time stamped with the local server's time.
- Use Greenwich Mean Time (GMT): each log entry is time stamped according to Greenwich Mean Time.

**Log cycle:** controls how often log files are closed and new log files created. You can select to have logs cycled hourly, daily, weekly, monthly, or turn cycling off. The default value is daily, which means that at midnight each day, all log files will be closed and new log files created. Log files cycle at midnight for all settings except hourly and off. This field is required. Directive: LogCycle

- Notes: When this option is set to "off", the log will not rollover, and log maintenance will not occur. The "weekly" value may not be set correctly if your server is not using the Gregorian calendar.

**Maximum log file size:** limits the size of any defined log file (in bytes, kilobytes, megabytes, or gigabytes). This value prevents unbounded log file growth. If a log file exceeds this size, no more information will be written to it. Valid values include 0 through 2,147,483,647 (bytes). The default value is 0 bytes, which means there is no log file size limit. If the default value of 0 bytes is used, the log files will grow without bound. The first drop down menu provides the options Bytes, Kilobytes, Megabytes, Gigabytes. The second drop down menu provides the options Minimum (1 byte), Maximum (2,147,483,647 bytes) and Unlimited (0 bytes). This field is optional. Directive: LogLength

# HTTP Server Trace



- Traces information about server operation
  - Process management, URI interpretation etc.
- Can be started by GUI and CL command



<input type="radio"/>	WEBJIMSVR1	Apache	Stopped	Global	*:80
<input checked="" type="radio"/>	WEBSERVER	Apache	Stopped	Global	10.1.100.100:8008
<input type="radio"/>	WEBWIZ	Original	Stopped	Yes	as20.itsoroch.ibm.com:8974

Server startup parameters:  ?

[Refresh](#) [Start](#) [Delete](#) [Rename](#) [Manage Details](#)

# Notes



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 screen by starting the server with the lowercase `-ve`, `-vi`, `-vv` startup parameters. The Start TCP/IP Server (STRTCP SVR) CL command also supports these startup options. The same data can also be collected 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`, `-vv` switches on server restart. If you are unable to end all server jobs use the Trace TCP/IP Application (TRCTCPAPP) command instead.

The HTTP server trace can be set to operate at three different levels called error, information, 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 will be dumped to file QAP0ZDMP in QTEMP in a member called QP0Znnnnnn (where nnnnnn is the HTTP server job number you fed to the DMPUSRTRC command). The following table illustrates the usage and purpose of this tracing facility.

Startup switch	TRCTCPAPP	Trace output
<code>-ve</code>	*ERROR	Server startup only. Nothing else is recorded unless an error occurs.
<code>-vi</code>	*INFO	Server startup and initialization, including directive processing, character conversion and client request handling.
<code>-vv</code>	*VERBOSE	All the above plus API and module invocation, HTTP headers, error messages.

# Other Startup Parameters



**These options are available as server startup parameters**

- -netccsid [nnn]
- -fscssid [nnn]
- -d [serverroot]
- -f [configuration]
- -C [directive]
- -c [directive]
- -V
- -l
- -t

# Notes



Server startup parameters can also be of aid in problem determination as well as in testing your sever configuration. In addition to the more debug-oriented `-ve`, `-vi`, `-vv` startup parameters for “HTTP server trace”, the following options are available:

- `-netccsid [nnn]` Overrides the `DefaultNetCCSID` directive.
- `-fscsid [nnn]` Overrides the default `DefaultFsCCSID` directive.
- `-d [serverroot]` Set the initial value for the `ServerRoot` variable to `serverroot`. This can be overridden by the `ServerRoot` directive.
- `-f [configuration]` Use the values in the configuration on startup. If the configuration does not begin with a `/`, then it is treated as a path relative to the `ServerRoot`.
- `-C [directive]` Process the given directive just as if it had been part of the configuration.
- `-c [directive]` Process the given directive after reading all the regular configuration files.
- `-V` Display the base version of the server, build date, and a list of compile time settings, then exit.
- `-l` Display a list of all modules compiled.
- `-t` Test the configuration file syntax but do not start the server. This command checks to see if all `DocumentRoot` entries exist and are directories.





# Highly Available Web Server Clusters

# Notes



This part covers these three Web server cluster models that are supported:

- ▶ HA Web Server: Major Features
- ▶ Primary/backup with takeover IP model
- ▶ Primary/backup with a network dispatcher model
- ▶ Peer model

# HA Web Server: Major Features



**Makes the IBM HTTP Server (powered by Apache) highly available across a cluster of iSeries nodes**

## **Support for different models:**

- Availability
  - Primary/Backup with IP-Takeover
  - Primary/Backup with IBM's e-Network Dispatcher
- Availability and Scalability
  - Peer with e-Network Dispatcher

## **Replication of HA CGI and Net.Data state data**

- CGI and Net.Data programs store Persistent CGI state in Web server
- State is then available on other nodes in cluster

## **Integration with Cluster Resource Services to detect failures**

- High-Availability Business Partner (HABP) software is optional

**Improved availability of individual Web servers!**

# Notes



# When do you need HA Web Servers?

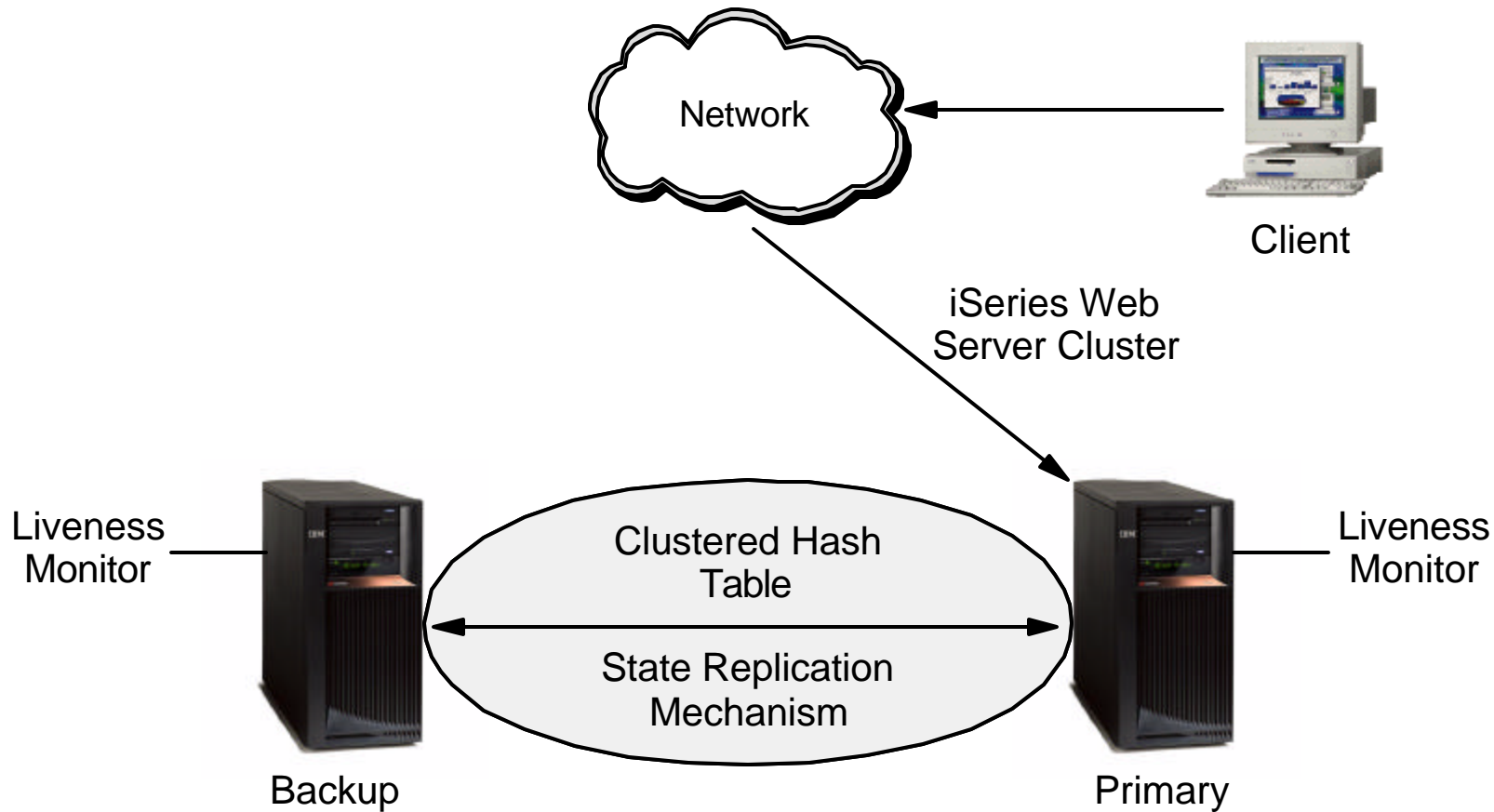


- If Web serving is a critical aspect of your business
- If you have periodic planned outages (e.g. for regular maintenance)
- If you want reasonable protection from unplanned outages
- If you want horizontal scalability of your web serving environment by distributing client requests across a multitude of server nodes

# Notes



# Takeover IP model



# Notes



## Primary/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 a 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.

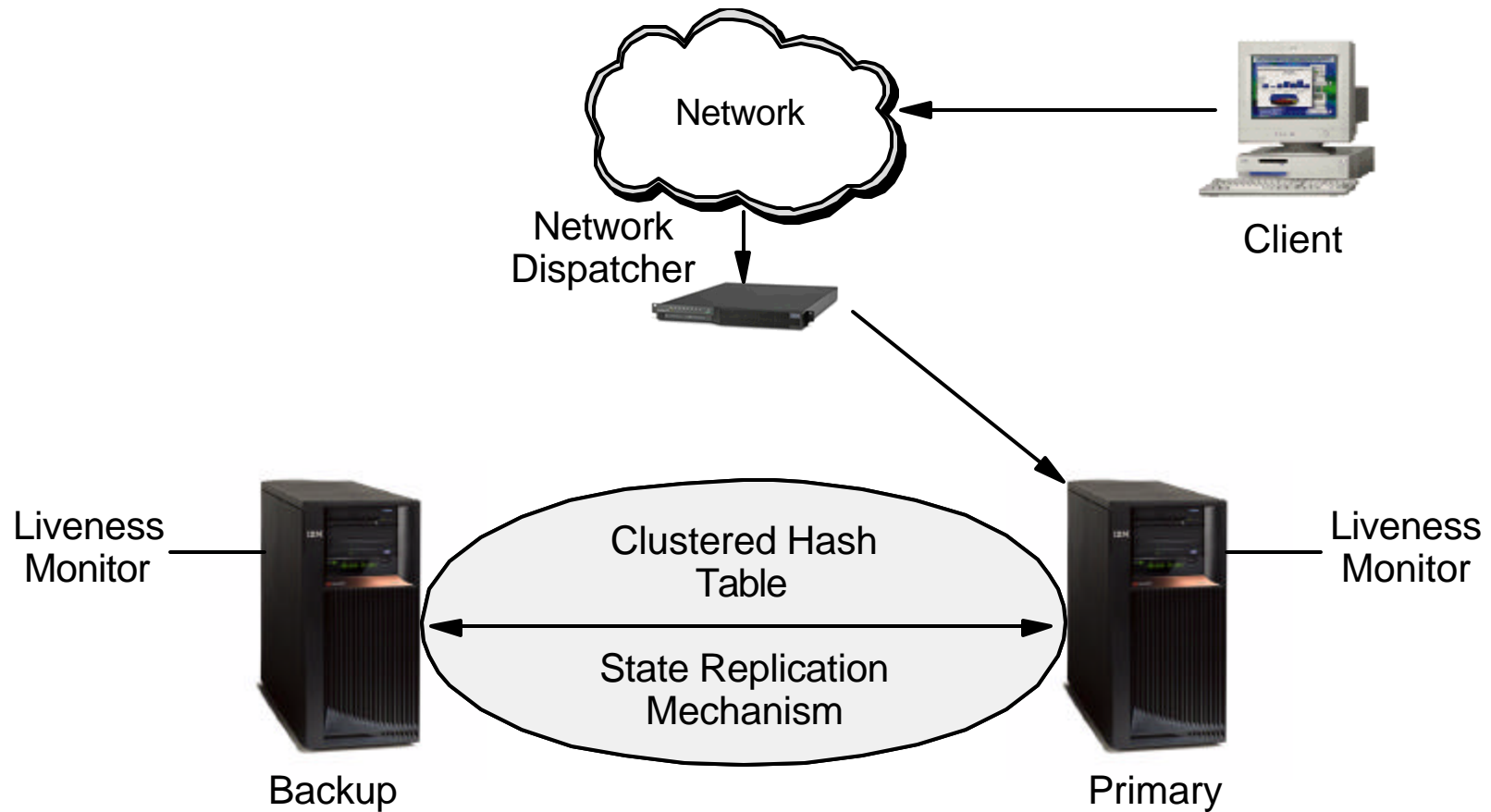
**Note:** For all the models presented here the CGI application has some extra work. On a single iSeries server (and before this HA was available) when a 'normal' persistent CGI application is done with one step of long-running session it and its static variables are held in a wait state - waiting for the client to respond. When this new HTTP request comes into the system the iSeries server will attach to the same persistent CGI job and away it will go (as it still knows about state due to the static variables that have not changed). - - - To operate in this HA environment on two different iSeries servers one thing changes: the CGI application - right when it is done with one step of long-running session - stores all its static variables (stores the state of the session) into the Clustered Hash Table. The Clustered Hash Table is then automatically made available on all systems operating in the clustered HTTP servers. When the next HTTP request comes in from the client the CGI job that is attached must read the static variables (retrieves the state of the session) from the Clustered Hash Table.

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. The client requests are redirected to the new primary node.
3. If the new primary receives a user request that belongs to a long-running-session (a CGI program that has been updated to be a highly available CGI program), the server will restore 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.
4. After the failed node recovers, the highly available Web server instance can be restarted and it will become the backup system. If the system administrator wants the failed node to become primary again, a manual switchover must be performed (this can be accomplished with the IBM Simple Cluster Management interface available through Operations Navigator or a business partner tool).



# Network Dispatcher Model



# Notes



## Primary/backup with a network dispatcher model

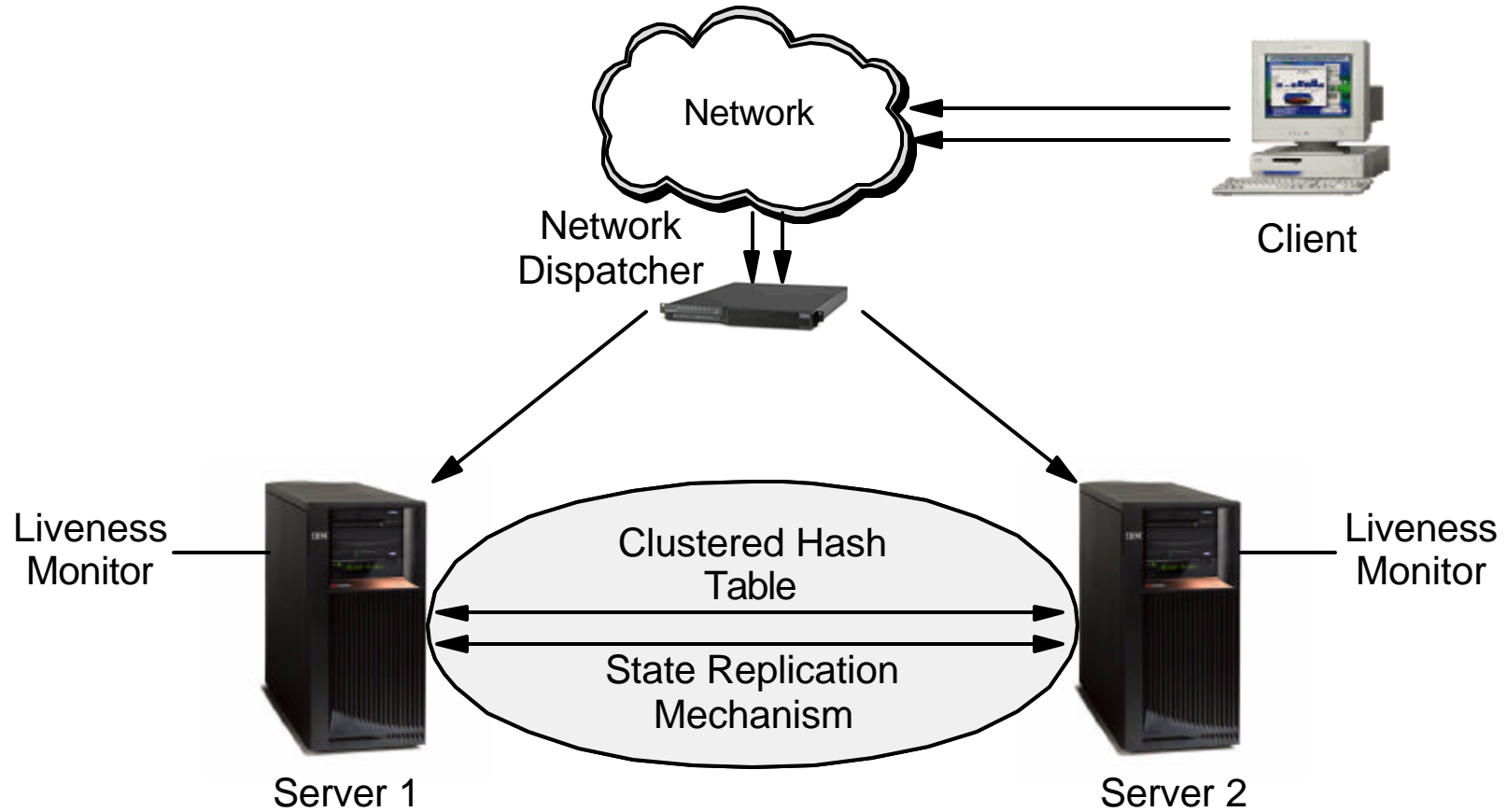
In this model, just like the primary/backup with takeover IP model, the Web server runs on the primary and all backup nodes. The backup nodes are in an idle state and all client requests are served by the primary node. A network dispatcher (for example the IBM WebSphere Edge Server) sends client requests 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 will become a backup Web server. If the system administrator wants the failed node to become primary again, a manual switchover must be performed.

**Note:** A node can join a recovery domain as primary only if the cluster resource group is in inactive mode.

# Peer model



# Notes



## 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 (for example the IBM WebSphere Edge Server) evenly distributes requests to different cluster nodes. This guarantees distribution of resources in case of heavy load. Linear scalability is not guaranteed beyond a small number of nodes. After some number of nodes are added, scalability can disappear, and the cluster performance can deteriorate.

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.

# Related Publications



- *The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this workshop.*

## International Technical Support Organization Publications

- For information on ordering ITSO publications, visit us at <http://www.redbooks.ibm.com> (Internet Web site) or
- <http://w3.itso.ibm.com> (intranet Web site)

For Technical Support see <http://www.ibm.com/support> and <http://w3.ibm.com/support>

## Redbooks on CD-ROMs

- Redbooks are available on CD-ROMs.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbook	SK2T-8038
AS/400 Redbooks Collection	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SK2T-8041
Application Development Redbooks Collection	SK2T-8037
Personal Systems Redbooks Collection	SK2T-8042

# Related Publications - Continued



## Other Publications

- *These publications are also relevant as further information sources:*

Title	Publication Number	
<p><a href="http://www-1.ibm.com/servers/eserver/series/software/http/">http://www-1.ibm.com/servers/eserver/series/software/http/</a> <i>APACHE SERVER UNLEASHED</i> <i>PROFESSIONAL APACHE</i> <i>APACHE SERVER ADMINISTRATOR's HANDBOOK</i> <i>APACHE SERVER BIBLE</i> <i>APACHE for DUMMIES</i> <i>APACHE: THE DEFINITIVE GUIDE</i> <i>IBM HTTP SERVER POWERED BY APACHE ON RS/6000</i> <a href="http://www-1.ibm.com/servers/eserver/series/education/">http://www-1.ibm.com/servers/eserver/series/education/</a></p>	<p>iSeries HTTP servers home by Bowen and Coar by Peter Wainright by Mohammed J. Kabir by Mohammed J. Kabir by Ken Coar by Ben Laurie, Peter Laurie SG24-5132 - IBM ITSO This presentation</p>	
<p>HTTP Server (powered by Apache) IBM iSeries Integration at its Best (send a note to <a href="mailto:brsmith@us.ibm.com">brsmith@us.ibm.com</a> to be added to a distribution list!)</p>	<p>SG24-6716 - IBM ITSO</p>	
<p>Redpaper: Bring PHP to Your</p>	<p>iSeries Server</p>	<p>REDP3639 - IBM ITSO</p>