



Standards and Practices

S&P Article ID: SAP-001

Title: Best Practices for Web Servers

Description: Configuration and management for web servers

Revision / Date: 1.1 / 2007-01-29

Status: Final

Author: Glenn Barnas

Client Approval:

Guidelines for Configuring and Supporting Multiple Web Server Instances

Proper consideration must be given to the design and implementation of web servers, especially those that will support multiple web sites. This document describes the technology and the recommended methods for implementing web server instances.

Terminology: Hardware Server / Software Server

The term “web server” usually refers to the physical hardware – CPU, Memory, Disk, and Operating System – the components you would associate with any typical server computer. The web server software, capable of supporting many sites on one physical server, usually takes a slightly different meaning. A software “web server” usually represents a specific IP address assigned to the server. One or more “web sites” can be defined within a single web server. To avoid confusion, this document will refer to the hardware as the *web server*, and to the software web site as an *instance*.

A web site *instance* is defined as a single web entity (www.somesite.com) and any subcomponents associated with that site. Instances can include subcomponents, known as *virtual directories* or *additional document directories* that allow mapping external data structures into a single web site.

Physical vs. Logical structure

Before any planning of your server can occur, you must understand the difference between physical and logical data structures. The *physical* structure defines how the data is arranged on the server’s hard disk. Usually, this is a collection of folders that help organize files, scripts, and data files for a particular web application. It does not have to – and in most cases, should not – mirror the actual configuration of the web site. The *logical* structure describes how the folders are arranged from the perspective of the root of the web site.

For example, a web site has a collection of files, along with some physical folders to group sections of the application files. It also has may have some *virtual* folders. Virtual folders are mapped by the web server service from a physical location to a logical location within a specific web site. This allows a single physical folder to be shared by all web sites (usually for common graphic images or scripts). It also allows specific security attributes to be applied, which would be different from the main (root) folder. The following table illustrates the relationship between physical and logical locations.

Physical	Logical	Security *
D:\webs\Site1\web	http://www.abc.com/	Read
D:\webs\Site1\scripts	http://www.abc.com/scripts	Read, Script, & Execute
D:\Apps\CFShare	http://www.abc.com/ColdFusion	Read & Script
D:\webs\Site2\web	http://www.xyz.com/	Read
D:\webs\Site2\scripts	http://www.xyz.com/scripts	Read, Script, & Execute
D:\Apps\CFShare	http://www.xyz.com/ColdFusion	Read & Script

* Logical permissions, set by the web server.

As you can see from the example, “*scripts*” is a virtual web folder with specific permissions, pointing to a folder specific to the application for Site1. The second example creates a virtual directory that is shared by all web sites to access ColdFusion scripts. Note that it is a different directory path from the base web data, and that the logical and physical names are different.

Server URL Mapping

When a physical server hosts more than one web site, some method of routing the user requests to the appropriate site is required. Three components are typically available to perform this function – IP Address, Port Number, and Host Header Name.

- **IP Address** – This is the most generic of routing methods. Any HTTP request sent to a specific IP address are delivered to that server. These requests are routed internally to the web server application. The web server instance may not necessarily have an IP address assigned to it. Thus, will accept any HTTP request not delivered to a more specific server instance.

When multiple IP addresses are assigned to a server, the web server application can also be assigned an IP address, allowing HTTP requests to be routed to that server instance.

- **Port Number** – Somewhat more specific than IP address routing alone. Often, a public web site will run using port 80, while the secure version of that site will operate on the same IP address, but use port 443. Each site is a separate server instance. Other port numbers (above 2048) can be used, but this is not usually convenient to the user. Port numbers are always used in conjunction with a specific IP address.
- **Host Header Name** – The most specific form of HTTP routing, used in conjunction with an IP address and a port number. The server actually examines the URL, extracting the DNS name and routing the request to a specific web server instance. Host headers cannot be used with secure (HTTPS) traffic.

Using the above concepts, one server can have multiple IP addresses, each IP address can have multiple ports, and each port can have multiple Host Headers assigned. Thus, many web sites can be hosted on a single server.

To use an analogy, imagine that the server (hardware) is a piece of real estate, big enough for several large buildings. Each IP address represents an apartment building, which can have several floors, and several apartments per floor. The port number is used to direct the request to a specific floor while the Host Header points to a specific apartment on that floor. As in real life, a rich tenant (priority application) can utilize an entire floor (port), or even an entire building (IP address).

Generally, the default document directory in the web server application does not have an IP address, Port, or Host Header assigned to it. This web server instance responds to all requests that cannot be handled by a specific site instance. Thus, if a web site is unavailable for any reason, the request is directed to the default web site instance (like the super’s office in an apartment complex). This can present a simple error message, or provide a menu with links to alternate sites.

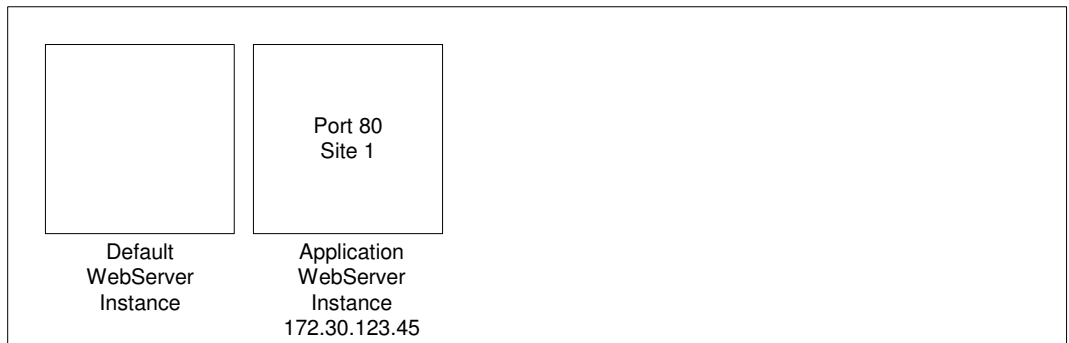
Refer to the following diagram, which illustrates the physical server, web server (software) instances, along with breakdowns by IP Address, Port #, and Host Header Name.

Web Server
POP000



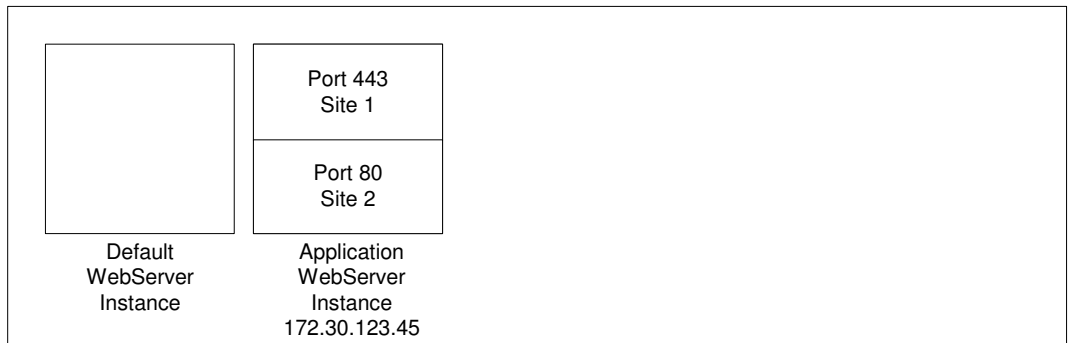
Server with single instance (default only)

Web Server
POP000



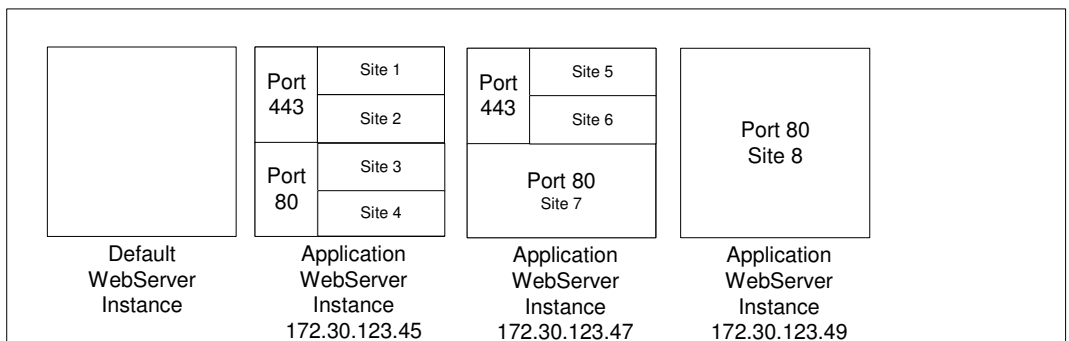
Server with single specific instance plus default site

Web Server
POP000



Server with multiple specific instances using ports to route requests, plus default site

Web Server
POP000



Server with multiple specific instances using combination of IP Address, ports & host header names, plus default site

Planning the Data Structure

Good organization of application data is imperative to smooth operation of a web server. By planning the structure, you can minimize the administrative overhead of creating directories, applying security, and similar tasks. By implementing a standard directory structure for both the physical storage and the logical web site, several important benefits are realized.

- A server can support multiple web sites without conflict between sites. Each site's data is maintained in separate physical directories, isolated from all other sites.
- Secure access can be given to each site's data, via network shares or FTP access.
- Utilizing a structured subdirectory arrangement, precise security settings can be independently applied to the web root folder and any other virtual folder in the logical web site structure.
- Placing all web data files in an external structure allows clean migration from one web server application to another (ie: iPlanet to Apache or IIS).

There are many different ways to organize your web data structure. One of the most common methods used in the industry is to organize the data by customer site, application, or hosting designer. Regardless of the organizational method chosen, it is crucial to define a standard and stick with it.

As we will illustrate, every site consists of a top-level folder and at least two subfolders. Each top-level folder represents a unique web site. This folder is shared (or mapped to an FTP folder) to permit remote management of the web site files by the appropriate remote support staff. The subfolders defined here represent, at a minimum, the logical root of the web site and the *scripts* subfolder. This allows different permissions, whether physical or logical, to be applied to these folders on an application-specific basis.

In a configuration where the server is dedicated to a specific customer and one or more application web sites are hosted, each application would be located in a subfolder of a common directory for all web data. This base directory – *webfiles* in the examples in this document – contains one subfolder for each web server instance. The development team might have share access to this base directory, and thus all application subfolders. The subfolders only serve to identify and isolate the applications from each other.

```
D:\WebFiles
- Application1
  - Scripts
  - WebRoot
- Application2
  - Scripts
  - WebRoot
- Common
  - Images
  - cgi-bin
```

Figure 2 – Single Customer Physical Data Structure

Any number of web site instances can be supported in this fashion, to the physical limits of the web server hardware and network connection.

For configurations where a physical server hosts applications from multiple customers, the configuration varies slightly. Each customer will now have a folder in the webfiles base path, with each of their applications being stored in a subfolder within the customer's main folder.

```
D:\WebFiles
- Customer 1
  - Application1
    - Scripts
    - WebRoot
  - Application2
    - Scripts
    - WebRoot
- Customer 2
  - Application1
    - Scripts
    - WebRoot
- Common
  - Images
  - cgi-bin
```

Figure 3 – Multi-Customer Physical Data Structure

In the above examples, the common folder contains subfolders that are mapped as virtual directories to multiple web site instances. This allows items such as logo and bullet images or utility scripts to be shared among all sites while minimizing maintenance and management issues that would occur if separate copies were maintained.

One significant advantage of this design is that it isolates the customer data and the web server. By maintaining the customer data external to the web server application folder, only the system administrator can make changes to the web server software. This prevents the customer from inadvertently making changes to a running web server without limiting what they can do to manage their own site data.

This design also allows multiple customers to each host multiple applications on a single server, utilizing O/S security to separate the customers and their applications from each other. This centralizes management, simplifies redundancy and fail-over, and reduces both hard and soft costs.

User Design Considerations

When planning the structure of a web server, the web manager must recognize the need for security, interaction with other applications, and understand that the web server can manipulate the physical server/disk structure to provide the desired logical web folder structure.

Let's review a sample site's requirements. Note that web (logical) paths use "/" and physical paths use "\" directory delimiters in this example. This may not represent the actual environment where the web server is installed (Windows vs. Unix).

- Access via URL `http://server.domain.org`
- Needs access to Cold Fusion scripts in "/CFScript" web folder. Physical path to CFScript is "D:\Cfusion\Bin\CFcgi-bin"
- Needs access to custom Java applets written by development team.
- Need access to 3rd party executables in "/3pscripts" web folder. This folder contains .EXE files as well as traditional web scripts (.asp, .jsp, and .cgi)
- Need to use FRIT or FRB standard logo images. These would generally be stored in a "/Common" web folder shared by all web instances on that server. The physical path is "\WebFiles\Common".
- Users of this application must be able to download data files using URLs to a FTP service. Data files must be maintained within the web application directory, but not accessible via HTTP protocol.

To accomplish this, the following configuration is defined:

- Create a folder structure `\WebFiles\AppName`, with subfolders `WebRoot`, `Scripts`, and `FTP`. Apply file-system permissions according to the example below:
WebRoot - Read=Everyone*,System; Write=Admins, Developers
Scripts - Read=Everyone*,System; Write=Admins, Developers
FTP - Read/Write=Everyone*
* Everyone can be replaced with "Authenticated Users"
- Place the application files (and any required subfolders) directly in the "WebRoot" folder. This folder is then defined as the root of the web server instance.
- Install the Java applets in `\WebFiles\AppName\Scripts`. Define this folder as a virtual folder "/Scripts" with Execute-only logical security.
- Move the 3pScripts folder to "`\WebFiles\AppName\3pScripts`" and apply appropriate physical security. Define this as a virtual web folder "/3pScripts" with Execute-only logical security.
- Place common images into the "`\WebFiles\Common`" folder. Define a virtual web folder "/common" that references this location, and define Read-Only logical security.
- Define a virtual web folder "/CFScripts" referencing the physical location, and define Execute-only logical security.
- FTP data files are stored in the "`\WebFiles\AppName\FTP`" folder. This folder is defined as a FTP root or virtual folder to the FTP server service. Apply Read or Read-Write logical permissions on the FTP server as appropriate.

Server Network Configuration

The server should be assigned a single IP address that represents the server itself. This IP should be used for general server management. An “A” record should be created to represent this IP address using the server’s designated host name.

Each web instance should be assigned a unique IP address. This IP address should be bound to the appropriate network adapter on the server. An “A” record should be created for each web instance IP using the primary server name, appended with “a” and a sequential number. A CNAME record should then be created to point to these additional addresses

For example, a server called WebProd00 will host two web instances. There will be three IP addresses bound on the server.

IP Address	Name	Use
10.32.16.8	webprod00	Primary IP address
10.32.16.97	webprod00a1	First application instance IP address
10.32.16.98	webprod00a2	First application instance IP address

Security Considerations

WebFiles Root Folder – This is the root of all web site content. Only administrators should have modify access to this folder, while the web anonymous account (IUSR_computer) should have read access. These permissions should be inherited by the child folders.

“WebApp” Folder – This is the folder that holds all content, including application code, for a specific web site/application. It is named so that it identifies the site or application.

Permissions should be inherited from the parent folder, and a local group used by the web administrators of this site should be created and granted Modify access. It is often helpful to make this folder a share, so admins/developers can access the folder without impacting other sites. This, of course, depends on local security policy regarding access to Dev, QA, and Production systems.

Application executables should be placed in sub-folders within this folder, but not in the WEB folder. Any subdirectories necessary to support the application may be created here.

Web Folder – All static content, as well as ASP type (ISAPI) scripts are maintained in this folder. Any folder structure can be created to meet the requirements of the web site.

Executables (.exe, .bat, .vbs, and dot.net) should not be placed anywhere in this folder structure – they should be placed in subfolders of the WebApp folder. Dot.net, dll, and exe files can be accessed via virtual folders. Batch and VB script files should never be accessible via an HTTP interface! If such scripts are needed by an application, they should be maintained in a separate folder under the WebApp folder